David C. Wyld
Dhinaharan Nagamalai (Eds)

Computer Science & Information Technology

6th International Conference on Image and Signal Processing (ISPR 2020)
June 20 ~ 21, 2020, Dubai, UAE

AIRCC Publishing Corporation

Volume Editors

David C. Wyld,
Southeastern Louisiana University, USA
E-mail: David.Wyld@selu.edu

Dhinaharan Nagamalai,
Wireilla Net Solutions, Australia
E-mail: dhinthia@yahoo.com

ISSN: 2231 - 5403
ISBN: 978-1-925953-21-3
DOI: 10.5121/csit.2020.100701- 10.5121/csit.2020.100704

Typesetting: Camera-ready by author, data conversion by NnN Net Solutions Private Ltd., Chennai, India

Preface

The 6th International Conference on Image and Signal Processing (ISPR 2020) June 20 ~ 21, 2020, Dubai, UAE, 6th International Conference on Artificial Intelligence (ARIN 2020), International Conference on Machine learning and Cloud Computing (MLCL 2020), International Conference on Networks, Blockchain and Internet of Things (NBIoT 2020) was collocated with 6th International Conference on Image and Signal Processing (ISPR 2020). The conferences attracted many local and international delegates, presenting a balanced mixture of intellect from the East and from the West.

The goal of this conference series is to bring together researchers and practitioners from academia and industry to focus on understanding computer science and information technology and to establish new collaborations in these areas. Authors are invited to contribute to the conference by submitting articles that illustrate research results, projects, survey work and industrial experiences describing significant advances in all areas of computer science and information technology.

The ISPR 2020, ARIN 2020, MLCl 2020 and NBIoT 2020 Committees rigorously invited submissions for many months from researchers, scientists, engineers, students and practitioners related to the relevant themes and tracks of the workshop. This effort guaranteed submissions from an unparalleled number of internationally recognized top-level researchers. All the submissions underwent a strenuous peer review process which comprised expert reviewers. These reviewers were selected from a talented pool of Technical Committee members and external reviewers on the basis of their expertise. The papers were then reviewed based on their contributions, technical content, originality and clarity. The entire process, which includes the submission, review and acceptance processes, was done electronically.

In closing, ISPR 2020, ARIN 2020, MLCl 2020 and NBIoT 2020 brought together researchers, scientists, engineers, students and practitioners to exchange and share their experiences, new ideas and research results in all aspects of the main workshop themes and tracks, and to discuss the practical challenges encountered and the solutions adopted. The book is organized as a collection of papers from the ISPR 2020, ARIN 2020, MLCl 2020 and NBIoT 2020.

We would like to thank the General and Program Chairs, organization staff, the members of the Technical Program Committees and external reviewers for their excellent and tireless work. We sincerely wish that all attendees benefited scientifically from the conference and wish them every success in their research. It is the humble wish of the conference organizers that the professional dialogue among the researchers, scientists, engineers, students and educators continues beyond the event and that the friendships and collaborations forged will linger and prosper for many years to come.

David C. Wyld
Dhinaharan Nagamalai (Eds)

General Chair — Organization

David C. Wyld, Southeastern Louisiana University, USA
Dhinaharan Nagamalai, Wireilla Net Solutions, Australia

Program Committee Members

Abdul Rasak Adegoke Zubair, University Of Ibadan, Nigeria
Ahmad T. Al-Taani, Yarmouk University, Jordan
Ahmet Cinar, Firat University, Turkey
Ali Salem, University of Sfax, Tunisia
Amel Ourici, University Badji Mokhtar Annaba, Algeria
Amin Karami, University of East London, UK
Ammar Almasr, Albalqa applied university, Jordan
Anas Shatnawi, University of Quebec at Montreal, Canada
Ashraf Hossain, National Institute of Technology (NIT), India
Atheer Yousif Oudah, Thi-Qar University, Iraq
Ayman Abu-Baker, Applied Science University, Jordan
Baghdad Atmani, University of Oran, Algeria
Bai Li, Woodside Energy Ltd, Australia
Bala Modi, Gombe State University, Nigeria
Barbaros Preveze, Cankaya University, Turkey
BENYETTOU Mohammed, Univesity Center of Relizane, Algeria
Bing Li, Xi'An Technological University, China
Boukari Nassim, Skikda University, Algeria
Braham Barkat, The petroleum Institute, Saudi Arabia
Chris T. Panagiotakopoulos, University of Patras, Greece
Chrissanthi Angeli, University of West Attica, Greece
Chukwuemeka Ifegwu Eke, University of Abuja, Nigeria
Dalila Guessoum, Saad Dahleb University, Algeria
Daniel Dacuma Dasig, Jose Rizal University, Philippines
Dongchen Li, Peking University, China
Ehsan Heidari, Islamic Azad University, Iran
Elena Pelican, Ovidius University of Constanta, Romania
Fatemeh Deregeh, Shahid Bahonar University of Kerman, Iran
Fatih Korkmaz, Cankiri karatekin University, Turkey
Francisco Macia Perez, University of Alicante, Spain
Gelenbe, Imperial College, UK
Glaoui hachemi, Tahri Mohammed University, Algeria
Goreti Marreiros, Polytechnic of Porto, Portugal
Grienggrai Rajchakit, Maejo University, Thailand
Guilherme Gomes, Federal University of Itajub, Brazil
Gurkana, Yildiz Technical University, Turkey
Hamid Ali Abed AL-Asadi, Iraq University college, Iraq
Hamid Mcheick, University of Quebec at Chicoutimi, Canada
Hatem Aldelaimi, Baghdad University, Iraq
Hatem Khater, Horus University, Egypt
Hatem Mohamed Abdelkader, Menofia University, Egypt
Hazem El-Gendy, Ahram Canadian University, Egypt
Héctor Migallón, Miguel Hernández University, Spain

Technically Sponsored by

Computer Science & Information Technology Community (CSITC)

Artificial Intelligence Community (AIC)

Soft Computing Community (SCC)

Digital Signal & Image Processing Community (DSIPC)

Organized By

Academy & Industry Research Collaboration Center (AIRCC)

TABLE OF CONTENTS

6th International Conference on Image and Signal Processing (ISPR 2020)

6th International Conference on Artificial Intelligence (ARIN 2020)

International Conference on Machine learning and Cloud Computing (MLCL 2020)

International Conference on Networks, Blockchain and Internet of Things (NBIoT 2020)

HAND SEGMENTATION FOR ARABIC SIGN LANGUAGE ALPHABET RECOGNITION

Ouiem Bchir

College of Computer and Information Sciences
Computer Science Department, King Saud University
Riyadh, Saudi Arabia

ABSTRACT

This research aims to separate the hands from the background of colored images representing the Arabic Sign language alphabet gestures. This hand segmentation task is one of the main challenges of image based Sign language recognition systems due to the issue of skin tones variations and the complexity of the background. For this purpose, an efficient system that segment the hand object and separate it from the rest of the image based on deep learning is investigated. More specifically, the DeepLab v3+ network architecture that is a combination of spatial pyramid pooling module and encode-decoder structure will be trained to learn the visual characteristics of the hand and segment it with detailed boundaries. The effectiveness of the proposed solution is investigated on a large dataset of size 12000 with an accuracy of 98%, an IoU of 93% of and BF score of 87%.

KEYWORDS

Hand segmentation, Deep learning, Arabic Sign Language Alphabet, spatial pyramid pooling.

1. INTRODUCTION

Sign language is an essential way of communication for hearing and speaking impaired persons. In fact, in addition to their speaking disability, they usually exhibit reading and writing skill deficiency. However, even though sign language allowed persons with this disability to communicate among them, few people without the hearing/speaking disability understand it. This excludes the persons with disability from communication yielding their isolation, and thus affects heavily their life and generates sentiments of loneliness and frustration. This communication issue can be alleviated by the translation of the gestures of the sign language to a text and verse versa, especially with the recent technological advancements that have helped the development of systems that support sign languages recognition [1] [2], and thus have yielded the apparition of several systems on sign language recognition [3]. These systems are either based on images or on sensors.

Sensor-based systems require the utilization of gadgets like gloves or captors that capture the location and the gesture shape of the hand [4]. The problem with these gadgets is that they are cumbersome and non-convenient for the user to ware or to use. On the other hand, image based approaches are non-intrusive [5]. It is an alternative solution that has the advantage to be available and natural since it does not constrain the hearing impaired person to use any gadget and can be deployed using smart devices using the availability of cameras on the portable devices.

David C. Wyld et al. (Eds): ISPR, ARIN, MLCL, NBIoT - 2020
pp. 01-09, 2020. CS & IT - CSCP 2020 DOI: 10.5121/csit.2020.100701

Typically, image-based systems separate the hand object from the other objects in the scene and recognize the corresponding gesture using image processing and machine learning techniques. More specifically, the captured image of the scene, is first segmented into two objects, the hand and the background. The hand object is then conveyed as input to the recognition system. However, appropriate features need to be extracted from the image in order to efficiently discriminate the hand from the background. This choice is challenging and is considered as an issue of image-based systems [6] due to the variability of the skin colors and the complexity of the backgrounds.

In this paper, the segmentation problem of the hand is addressed. For this purpose, an efficient system that segment the hand object and separate it from the rest of the image for Arabic sign language alphabet recognition is investigated. The proposed approach is based on deep learning. Namely, the DeepLab v3+ network architecture [7] will be trained to learn the visual characteristics of the hand and segment it.

2. RELATED WORKS

Hand segmentation is the first task of image based sign language recognition system. However, it is not a straightforward task due the skin color variations and the complexity of the backgrounds. An interesting characteristic used for segmenting the hand is the skin color, since it is not sensitive to scale, location, and orientation of the hand. For this reason, several approaches used the pixel values to construct the hand model by employing parametric models such as Gaussian Mixture Model (GMM), and non- parametric models such as histogram based methods. Nevertheless, these color based models are sensitive to illumination and skin tone variations. To alleviate this issue, the chrominance components are usually used instead of the pixel color neglecting this way the luminance components. Nevertheless, there are some challenges that still restrict the segmentation process which are complexity of the background, and the low quality of the image.

Existing Arabic Sign language recognition systems addressed the segmentation problem in different ways. In fact, some approaches ([8],[9],[10],[11]) circumvent the segmentation task by restraining the gesture images to have a uniform background. Other reported works ([12] [13]) use depth cameras, such as Kinect, and segment the hand as the nearest object to the camera. On the other hand, other approaches made the segmentation problem easier by using gadgets such as colored gloves [14]. However, these approaches ([12] [13] [14]) are inconvenient due to the unavailability and the cost of the sensor or the cumbersomeness of the gloves.

Hand segmentation for Arabic Sign language alphabet recognition based on skin pixel's characteristics have been reported in the literature. The authors in [15] convert the RGB input image to the YCbCr space and then use the skin profile to separate the hand. To further enhance the segmentation results, morphological dilation [16] is performed. Similarly, for the purpose of hand segmentation for Arabic Sign language, the authors in [17] convert the RGB input image to the YCbCr space. On the obtained color space, the Grey Level Co-occurrence Matrix (GLCM) [16] is computed. Using the GLCM values, the contrast, the correlation, the energy, and the local homogeneity are calculated and conveyed to a Multiple Layer Perceptron classifier [18] in order to recognize the skin. On the other hand, the authors in [19] segment the hand using the iterative thresholding algorithm [16] after de-noising the input image using the median filter. Although the approach in [12] use depth cameras and the hand is segmented as the closest object using the depth information, they also perform pixel segmentation to deal with complex backgrounds. For this purpose, they use the RGB ratio model. It is defined as in (1)

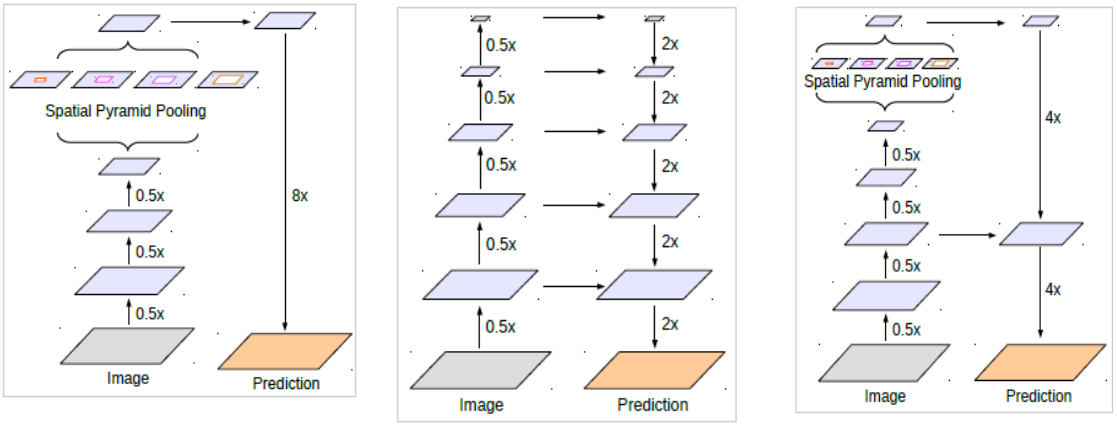
$$s = \begin{cases} 1 & if\ 0 \leq \frac{R-G}{R+G} \leq 0.5\ and\ \frac{R}{R+G} \leq 0.5 \\ 0 & otherwise \end{cases} \quad (1)$$

where s is the intensity of the segmented image, and R and G are the red and green color component of the input image.

As mentioned above, one of the main challenges of segmenting the hand is the choice of the suitable feature that allow discriminating the hand from the background. Deep learning based approaches can omit this choice since the feature is learned through the network architecture. However, Arabic Sign languages systems based on deep learning that have been reported so far ([20] [14] [13]) don't address the segmentation problem. They either consider hand images with uniform backgrounds, use depth sensors or gadgets. In summary, the hand segmentation for sign language recognition is still considered as an open issue problem.

Semantic segmentation that aims to predict a semantic label to every pixel [23] [25] is one of challenging tasks in computer vision. In this context of semantic segmentation, Fully Convolutional Networks (FCNs) [21,22] have shown promising results [23,24]. For this reason, various models have been proposed to extract the visual information [25, 26]. They include network models with multi-scale inputs [27, 28] or probabilistic graphical models like DenseCRF [17]. Besides, model networks such as PSPNet [29] or DeepLab [28, 7] that perform spatial pyramid pooling [30, 31] at several grid scales or apply atrous convolutions with different rates have demonstrated significant result improvement on several segmentation problems.

Recently, the DeepLabv3 model has been improved by combining the spatial pyramid pooling module (Figure1. (a)), with the encoder-decoder structure (Figure1. (b)) [32]. This results in the DeepLabv3+ model which learns rich semantic information from the encoder, and detailed object boundaries by the decoder. As shown in figure 1. (c), the encoder module extracts features at different resolutions by performing atrous convolution.



(a) (b) (c)

Figure 1. Combination of the spatial pyramid pooling module with the encoder-decoder structure, (a) Spatial Pyramid Pooling, (b) Encoder-Decoder, (c) Encoder-Decoder with Atrous Conv [32].

3. HAND SEGMENTATION SYSTEM

This research intends to segment the hand for Arabic sign language alphabet recognition using the DeepLabv3+ deep learning architecture. As in [32], two neural network modules are considered. Namely, spatial pyramid pooling module [33] and encoder-decoder structure [34] are used. The spatial pyramid pooling module is responsible for extracting the hand features and the encoder – decoder network learns the hand boundaries. In fact, even though the spatial pyramid pooling module encodes the visual information of the hand in the last layer, the hand boundary information is not available due to the pooling operation and the convolution striding through the network. A denser feature mapping can address the problem but it would be computationally prohibitive while the encoder-decoder network is fast. For this reason, as in [32], the two architectures are combined where the encoder module incorporates the feature extraction task. More specifically, we use the DeepLabv3+ architecture [32] that adds a decoder module that recovers the hand boundaries. Therefore, the extracted feature is encoded in the output of the spatial pyramid pooling module, and the hand boundaries are provided by the decoder module. The system is depicted in figure 2. As shown, it consists into two main components: the encoder and the decoder.

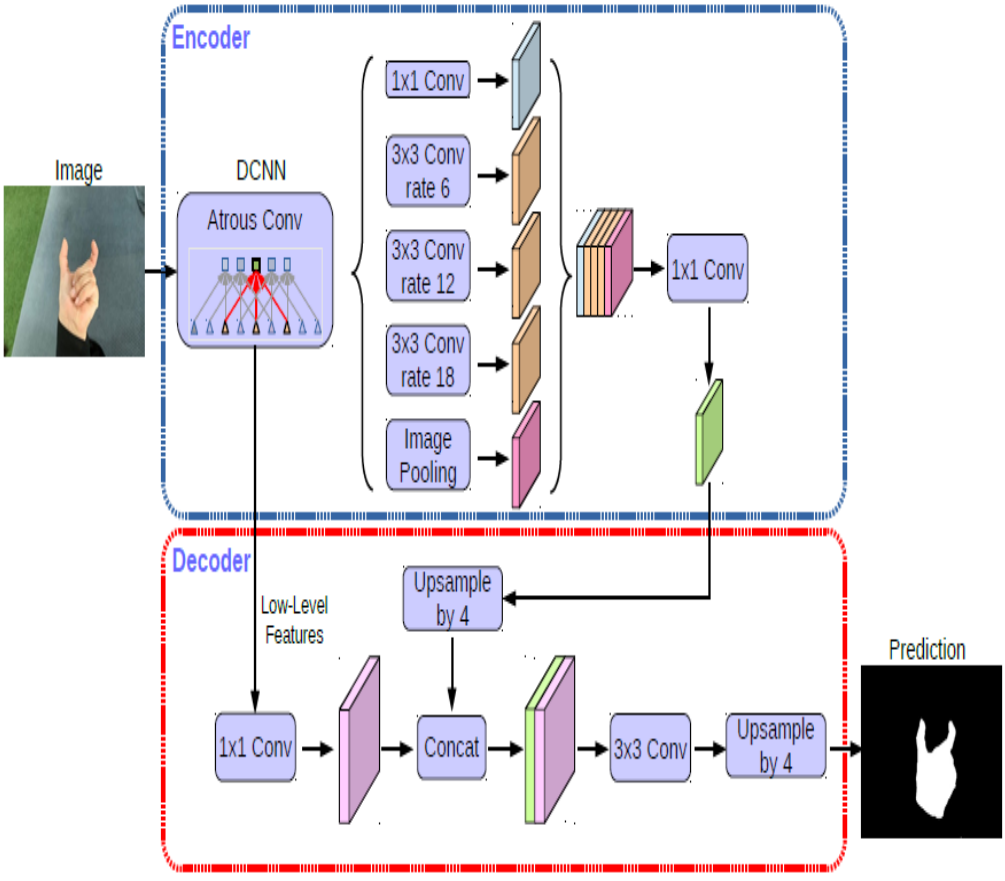


Figure 2. Segmentation system based on DeepLabv3+architecture [32].

The encoder module uses spatial pyramid pooling module [33] and employs atrous convolution [35] with different rates to extract the features. This results in a 256 coded vector that incorporate the semantic information of the hand at the output of the encoder. The decoder module up-

sample the output of the encoder bi-linearly by a factor of 4. The obtained vector is then concatenated to the convoluted low level feature. The concatenated feature is then convoluted with 3X3 filter and up-sampled by a factor of 4.

4. EXPERIMENTS

In order to assess the performance of the proposed system, a dataset of RGB images representing the 30 Arabic Sign Language gestures captured using mobile cameras is collected. Figure 3 displays samples from the dataset. The dataset includes 12000 images collected from 40 signers where each signer gestured the 30 letters in 10 different scenes. In order to construct the ground truth, the images were segmented manually where the hand is colored in white while the background is colored in black. The overall dataset is divided into 60% for training, 20% for validation, and 20% for testing.

Figure 3. Sample images representing the 30 gesture alphabets of the Arabic Sign Language alphabet.

The Deeplab v3+ network is setup using pre-trained Resnet-18 network for weight initialization. Table 1 reports the parameter's setting. Using the 60 % of the data and their corresponding ground truth images, the Network is trained. Then using the 20% of the validation data, the hyper-parameters are tuned. Finally, the system is tested using the remaining 20% of the data.

Table1. Parameter's setting

Learning rate	0.3
Learning period drop	10
Momentum	0.9
L2_Regularization	0.005
Max Epochs	30
Mini Batch Size	8
Shuffle	every epoch
Validation Patience	4

In order to evaluate the quality of the hand segmentation results against the ground truth, three performance measures are used, namely the accuracy, the Intersection over Union (IoU), also known as Jaccard similarity coefficient, and the BF Score. In order to compute these measure, the hand is considered as the positive class while the background is considered as the negative class. Thus, the number of true positives (TP) is the number of pixels belonging to the ground truth hand class and correctly segmented, and the number of false negative (FN) is the number of pixels belonging to the ground truth hand class and wrongly segmented as background. Similarly, the number of true negative (TN) is the number of pixels belonging to the ground truth background class and correctly segmented while the number of false negative (FP) is the number of pixels belonging to the ground truth background class and wrongly segmented as hand. The accuracy is then defined as in (2)

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

Since IoU is defined as the area of Intersection between the predicted hand and the hand in the ground truth divided over their union area, it can be expressed as in (3)

$$IoU = \frac{TP}{TP + FP + FN} \tag{3}$$

The BF Score which measures how close the learned hand boundary matches the ground truth hand is calculated as in (4)

$$BF = \frac{2 * Precision * Recall}{Precision + Recall} \tag{4}$$

where

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

and,

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

Using the three performance measures mentioned above, the quality of the hand segmentation results is evaluated against the ground truth. Table 2 reports the obtained results. As it can be seen, the segmentation approach performed very well on the considered Arabic sign language dataset. Figure 4 shows two examples that are correctly segmented.

Table 2. Performance results of the segmentation approach

Accuracy	IoU	BF Score
0.97941	0.92984	0.86562

In order to further investigate the obtained results, in Figure 5 hand segmentation examples where the segmentation was not correct are displayed. These bad result examples are related to the background characteristics. In fact, although the hand is detected, additional pixels from the background are wrongly predicted as hand. As it can be seen from Figure 5, this happens when two facts co-occur. The first one is that a part of the background color is similar to the hand skin color and the second one is that the overall region, constituted of the part of the background with similar color and the part of ground truth hand, has similar shape to another shape of the Arabic Sign language alphabet.

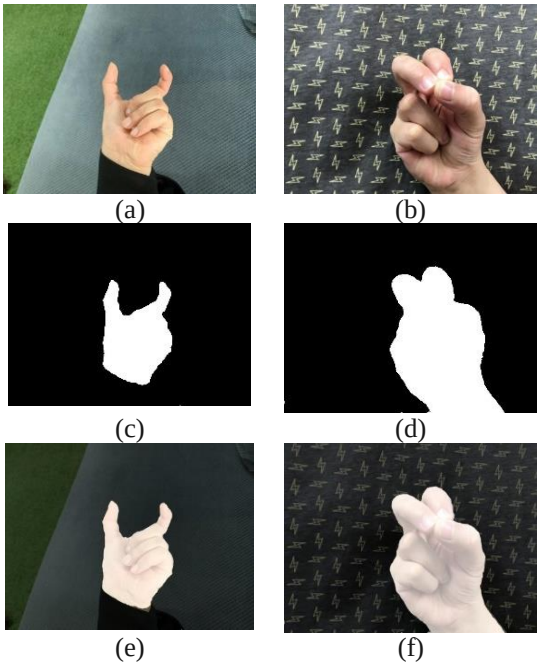


Figure 4. Two correctly segmented examples. (a) the first example of hand image, (b) the second example of the hand image, (c) the obtained segmentation of the image in (a), (d) the obtained segmentation of the image in (b), (e) the obtained segmentation over the original image in (a), (d) the obtained segmentation over the original image in (b).

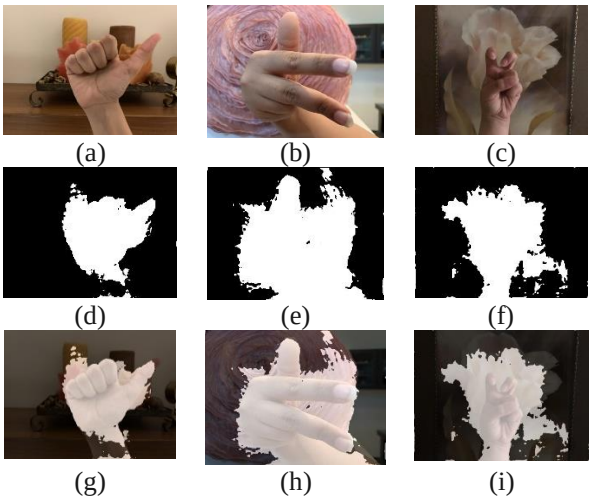


Figure 5. Three bad segmentation results. (a) the first example of hand image, (b) the second example of the hand image, (c) the third example of the hand image (d) the obtained segmentation of the image in (a), (e) the obtained segmentation of the image in (b), (f) the obtained segmentation of the image in (c), (g) the obtained segmentation over the original image in (a), (h) the obtained segmentation over the original image in (b), (i) the obtained segmentation over the original image in (c).

5. Conclusions and Future Works

In this paper, the DeepLabv3+ deep learning architecture is investigated for the segmentation of the hand pixels from images representing the Arabic Sign Language alphabets. The two modules of DeepLabv3+ allow effective semantic segmentation of the hand regions. In fact, the spatial pyramid pooling deep learning module foregoes the feature extraction stage as it is efficiently performed within the convolutional network. Moreover, the decoder module refines the segmentation results along the hand boundaries. The experimental results show the effectiveness of the proposed approach. As future works, the use of pre-trained networks other than Resnet-18 such as MobileNet v2 or ResNet-50 can be considered. Moreover, the obtained segmented images can be conveyed as input to an Arabic sign language alphabet recognition system.

References

[1] Tolba, M.F. and Elons, A. S., (2013) " Recent developments in sign language recognition systems", ICCES.

[2] Eisenstein, J., Ghandeharizadeh, S., Huang, L., Shahabi, C., G. Shanbhag, G., and Zimmermann, R., (2001) " Analysis of clustering techniques to detect hand signs", ISIMP.

[3] Brashear, H., Henderson, V., Park, K., , H., Lee, S. , and Starner, T., (2006) "American Sign Language Recognition in Game Development for Deaf Children", ASSETS, Portland, Oregon, USA.

[4] Mahesh, K., Mahishi, S., Pujari, S. R, N. S, and V., (2009) " Finger Detection for Sign Language Recognition", IMECS, Hong Kong

[5] Kang, S. K., Chung, K. Y., Rim, K. W., and Lee, J. H., (2011) " Development of Real-Time Gesture Recognition System Using Visual Interaction", IT Convergence and Security. Lecture Notes in Electrical Engineering, vol 120. Springer, Dordrecht

[6] Suharjito, F., Wiryana, F., Kusuma, G. P., and Zahra, A., (2018) " Feature Extraction Methods in Sign Language Recognition System: A Literature Review", INAPR .

[7] Chen, L.C., Papandreou, G., Schroff, F., Adam, H., (2017) " Rethinking atrous convolution for semantic image segmentation", arXiv.

[8] El-Bendary, N., Zawbaa, H. M., Daoud, M.S., Hassanien, A. E., and Nakamatsu, K., (2010) "ArSLAT: Arabic Sign Language Alphabets Translator",CISIM, Krackow, Poland.

[9] Sadeddine, K., Djeradi, R., Chelali, F. Z. and Djeradi, A., (2018) " Recognition of Static Hand Gesture", ICMCS.

[10] Tharwat, A., Gaber, T., Hassanien, A. E., Shahin, M. K., and Refaat, B., (2015) " SIFT-Based Arabic Sign Language Recognition System", Afro-European Conference for Industrial Advancement, Cham, 2015, pp. 359–370.
[11] Alzohairi R., Alghonaim, R., Alshehri, W., Aloqeely, S., and Bchir, O., (2018) " Image based Arabic Sign Language Recognition System", IJACSA, vol. 9, no. 3, 2018.
[12] Hamed, A. Belal, N. A. and Mahar K. M., (2016) "Arabic Sign Language Alphabet Recognition Based on HOG-PCA Using Microsoft Kinect in Complex Backgrounds",IACC, pp. 451–458.
[13] Aly, S., Osman, B., Aly, W. , and Saber M., (2016) " Arabic sign language fingerspelling recognition from depth and intensity images", ICENCO, Cairo, Egypt, 2016, pp. 99–104.
[14] Maraqa M. and Abu-Zaiter R., (2008) " Recognition of Arabic Sign Language (ArSL) using recurrent neural networks", ICADIWT, pp. 478–481.
[15] Hemayed E. E., and Hassanien, A. S., (2010) "Edge-based recognizer for Arabic sign language alphabet (ArS2V-Arabic sign to voice) ", ICENCO, pp. 121–127.
[16] Velho, L., Frery, A. C., and Gomes, J., (2009) ", Image Processing for Computer Graphics and Vision. Springer Science & Business Media.
[17] Dahmani, D., and Larabi, S., (2014) " User-independent system for sign language finger spelling recognition", Journal of Visual Communication and Image Representation, vol. 25, no. 5, pp. 1240–1250.
[18] Rosenblatt, F. F., (1963) "Principles Of Neurodynamics", Perceptrons And The Theory Of Brain Mechanisms.
[19] Al-Jarrah O. and Halawani, A., (2001) " Recognition of gestures in Arabic sign language using neuro-fuzzy systems", Artificial Intelligence, vol. 133, no. 1, pp. 117–138, Dec. 2001.
[20] Hayani, S., Benaddy, M., El Meslouhi, O. and Kardouchi, M., (2019) " Arab Sign language Recognition with Convolutional Neural Networks", ICCSRE, Agadir, Morocco, pp. 1–4.
[21] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., LeCun, Y. Overfeat. (2014) "Integrated recognition, localization and detection using convolutional networks",ICLR, 2014.
[22] Long, J., Shelhamer, E., Darrell, T., (2015) "Fully convolutional networks for semantic segmentation. CVPR, 2015.
[23] Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A., (2017) "Scene parsingthrough ade20k dataset", CVPR.
[24] Caesar, H., Uijlings, J., Ferrari, V., (2018) "COCO-Stuff: Thing and stuff classes in context", CVPR.
[25] Mostajabi, M., Yadollahpour, P., Shakhnarovich, G., (2015) "Feedforward semantic segmentation with zoom-out features", CVPR.
[26] Dai, J., He, K., Sun, J., (2015) " Convolutional feature masking for joint object and stuff segmentation", CVPR.
[27] Farabet, C., Couprie, C., Najman, L., LeCun, Y., (2013) "Learning hierarchical features for scene labeling", PAMI.
[28] Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L., (2017) " Deeplab:Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs",TPAMI.
[29] Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J., (2017) "Pyramid scene parsing network", CVPR.
[30] Grauman, K., Darrell, T. (2005) " The pyramid match kernel, Discriminative classification with sets of image features",ICCV.
[31] Lazebnik, S., Schmid, C., Ponce, J., (2006) "Beyond bags of features, Spatial pyramid matching for recognizing natural scene categories", CVPR.
[32] Chen, L. C. et al., (2018) "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation", ECC.
[33] He, K., Zhang, X., Ren, S., Sun, J. ,(2014) "Spatial pyramid pooling in deep convolutional networks for visual recognition", ECCV.
[34] Badrinarayanan, V., Kendall, A., Cipolla, R. Segnet., (2017) "A deep convolutional encoder-decoder architecture for image segmentation",PAMI.
[35] Papandreou, G., Kokkinos, I., Savalle, P.A., (2015) "Modeling local and global deformations in deep learning Epitomic convolution, multiple instance learning, and sliding window detection", CVPR.

BACK-PROPAGATION NEURAL NETWORK-BASED METHOD FOR PREDICTING THE INTERVAL NATURAL FREQUENCIES OF STRUCTURES WITH UNCERTAIN-BUT-BOUNDED PARAMETERS

Pengbo Wang[1*], Wenting Jiang[2] and Qinghe Shi[3]

[1]Institute of Solid Mechanics, Beihang University, Beijing, 100191, China
[2]Institute of Engineering Thermophysics, Chinese Academy of Sciences, Beijing, 100190, China
[3]School of Materials and Engineering, Jiangsu University of Technology, Changzhou, Jiangsu, 213001, China

ABSTRACT

Uncertain-but-bounded parameters have a significant impact on the natural frequencies of structures, and it is necessary to study their inherent relationship. However, their relationship is generally nonlinear and thus very complicated. Taking advantage of the strong non-linear mapping ability and high computational efficiency of BP neural networks, namely the error back-propagation neural networks, a BP neural network-based method is proposed to predict the interval natural frequencies of structures with uncertain-but-bounded parameters. To demonstrate the proposed method's feasibility, a numerical example is tested. The lower and upper frequency bounds obtained using the proposed approach are compared with those obtained using the interval-based perturbation method, which is a commonly used method for problems with uncertainties. A Monte Carlo simulation is also conducted because it is always referred to as a reference solution for problems related to uncertainties. It can be observed that as the varying ranges of uncertain parameters become larger, the accuracy of the perturbation method deteriorates remarkably, but the proposed method still maintains a high level of accuracy. This study not only puts forward a novel approach for predicting the interval natural frequencies but also exhibits the broad application prospect of BP neural networks for solving problems with uncertainties.

KEYWORDS

Back-propagation neural network, Natural frequency, Interval parameter, Perturbation method, Monte Carlo simulation.

1. INTRODUCTION

Natural frequencies have strong effects on the dynamic behaviours of structures and are of considerable importance for analysis, design, and optimization. However, uncertainty is ubiquitous in scientific research and engineering applications, and all structural analyses and designs involve varying degrees of uncertainty [1-3]. Owing to manufacturing deviations, modelling approximations, measurement inaccuracies, and external environmental changes, we may encounter many types of information uncertainty, such as external loads, material

David C. Wyld et al. (Eds): ISPR, ARIN, MLCL, NBIoT - 2020
pp. 11-31, 2020. CS & IT - CSCP 2020 DOI: 10.5121/csit.2020.100702

properties, geometric dimensions, and boundary conditions [4-6]. Thus, the natural frequencies of a structure are affected significantly by uncertainties. A commonly used approach to deal with an uncertainty problem is to model these uncertain parameters as random variables, and the resolution process is conducted using the probabilistic theory [7]. However, information about the probabilistic distributions of random variables may be unavailable or sometimes inaccurate, and often a small error in probabilistic information may lead to serious errors in probabilistic results [8,9]. Therefore, the probabilistic approach has some intrinsic limitations.

Recently, many scholars have shifted from probabilistic theory to interval theory to deal with uncertainty problems. In interval theory, all uncertain-but-bounded parameters can be described using interval numbers [10-13]. Interval theory has been applied to structural analysis and many papers have been published regarding this approach [14-17]. An interval-based perturbation method is commonly used to deal with uncertainties and has been successfully employed in the prediction of the interval natural frequencies of structures with uncertain-but-bounded parameters [18-20]. The perturbation method is highly accurate for linear problems, but its accuracy is poor for nonlinear problems. The mathematical relationship between the uncertain parameters and the natural frequencies usually remains nonlinear. The perturbation method only remains at a high level of accuracy when the varying ranges of interval parameters are small. As the varying ranges of interval parameters become larger, the errors generated by the perturbation method also increase rapidly. To obtain an accurate interval of natural frequencies with wide varying ranges of interval parameters, we focus on the application of artificial neural networks.

Artificial neural networks possess excellent information-handling capacity and have become powerful tools to study nonlinear systems [21]. Among all types of networks, the BP neural network, namely the error back-propagation neural network, is the most maturely studied neural network and has been successfully employed in structural analysis. Xu and Zhang used a BP neural network to implement a safety assessment for a bridge crane structure [22]. Yuan et al. employed a BP neural network to optimize the static and dynamic characteristics of machine tool structures [23]. Yang et al. studied the application of BP neural networks for hydraulic metal structure health diagnosing [24]. Li et al. used a BP neural network to predict the mechanical properties of shape memory alloy [25]. Although many scholars have aimed attention at the adoption of BP neural networks for structural analysis with deterministic parameters, the research of BP neural networks for structural analysis with uncertain parameters still remains at the initial stage, and very few papers have been published. In this study, we take advantage of the BP neural network properties of strong non-linear mapping ability and high computational efficiency and propose a BP neural network-based method to predict the interval natural frequencies of structures with uncertain-but-bounded parameters.

The remainder of this paper is organized as follows. In Section 2, the problem of predicting the lower and upper bounds of natural frequencies for structures with interval parameters is briefly introduced. In Section 3, the mechanism of a BP neural network is presented. In Section 4, the procedures of a BP neural network-based method are proposed. In Section 5, a numerical example is provided to illustrate the accuracy and efficiency of the proposed method. Section 6 presents the conclusions of this study.

2. PROBLEM FORMULATION

For a general engineering structure with n degrees of freedom, the governing equation for eigenvalue problems is given by:

$$\boldsymbol{Ku} = \lambda \boldsymbol{Mu} \tag{1}$$

where $\boldsymbol{K} = (k_{ij}) \in R^{n \times n}$ and $\boldsymbol{M} = (m_{ij}) \in R^{n \times n}$ are the stiffness and mass matrices, respectively; λ represents the eigenvalue and $\boldsymbol{u}$ denotes the eigenvector. The natural frequency corresponding to λ is: $f = \sqrt{\lambda}/2\pi$.

Taking uncertainty into consideration, the stiffness and mass matrices are subjected to the following constraints

$$\underline{\boldsymbol{K}} \le \boldsymbol{K} \le \bar{\boldsymbol{K}}, \quad \underline{\boldsymbol{M}} \le \boldsymbol{M} \le \bar{\boldsymbol{M}} \tag{2}$$

where $\underline{\boldsymbol{K}}$ and $\bar{\boldsymbol{K}}$ represent the lower and upper bounds of the stiffness matrix, respectively, and $\underline{\boldsymbol{M}}$ and $\bar{\boldsymbol{M}}$ are the lower and upper bounds of the mass matrix, respectively.
We define the nominal value matrices as

$$\boldsymbol{K}^c = (\underline{\boldsymbol{K}} + \bar{\boldsymbol{K}})/2, \quad \boldsymbol{M}^c = (\underline{\boldsymbol{M}} + \bar{\boldsymbol{M}})/2 \tag{3}$$

and the deviation amplitude matrices as

$$\Delta\boldsymbol{K} = (\bar{\boldsymbol{K}} - \underline{\boldsymbol{K}})/2, \quad \Delta\boldsymbol{M} = (\bar{\boldsymbol{M}} - \underline{\boldsymbol{M}})/2 \tag{4}$$

By means of interval matrix notation, the inequalities in Eq. (2) can be noted as

$$\boldsymbol{K} \in \boldsymbol{K}^I = [\boldsymbol{K}^c - \Delta\boldsymbol{K}, \boldsymbol{K}^c + \Delta\boldsymbol{K}], \quad \boldsymbol{M} \in \boldsymbol{M}^I = [\boldsymbol{M}^c - \Delta\boldsymbol{M}, \boldsymbol{M}^c + \Delta\boldsymbol{M}] \tag{5}$$

where $\boldsymbol{K}^I$ and $\boldsymbol{M}^I$ are the interval matrices.

When the stiffness matrix and mass matrix are assigned their nominal values, the nominal eigenvalues and the nominal eigenvectors can be obtained as

$$\boldsymbol{K}^c \boldsymbol{u}_i^c = \lambda_i^c \boldsymbol{M}^c \boldsymbol{u}_i^c, \ (i = 1, 2, 3, ..., n) \tag{6}$$

where λ_i^c is the ith order nominal eigenvalue and $\boldsymbol{u}_i^c$ is the ith order nominal eigenvector.

It can be inferred from Eq. (1) that if the matrices $\boldsymbol{K}$, $\boldsymbol{M}$ involve uncertainties, the eigenvalues and the eigenvectors are also uncertain. The eigenvalues with uncertainties can also be expressed with interval notation as follows:

$$\lambda_i = \lambda_i^c + \Delta\lambda_i, \quad (i = 1, 2, 3, ..., n) \tag{7}$$

where $\Delta\lambda_i$ represents the uncertain part of the i^{th} order eigenvalue λ_i .

An interval-based perturbation method is a commonly used method to deal with problems related to uncertainties, and it has already been introduced for the prediction of lower and upper bounds of eigenvalues for structures with uncertain parameters. The uncertain part of the eigenvalues can be calculated using the perturbation method as follows:

$$\Delta\lambda_i = (\boldsymbol{u}_i^c)^T \cdot \Delta\boldsymbol{K} \cdot \boldsymbol{u}_i^c - \lambda_i^c \cdot (\boldsymbol{u}_i^c)^T \cdot \Delta\boldsymbol{M} \cdot \boldsymbol{u}_i^c, \ (i = 1, 2, 3, ..., n) \tag{8}$$

and the interval eigenvalue λ_i^I can be expressed as:

$$\lambda_i^I = [\lambda_i^c - \Delta\lambda_i \ , \ \lambda_i^c + \Delta\lambda_i \ ], \quad (i = 1, 2, 3, ..., n) \tag{9}$$

For an interval parameter $a^I = [\underline{a}, \ \overline{a}] = [a^c - \Delta a, \ a^c + \Delta a]$, the percent change β is defined as:

$$\beta = \Delta a / a^c \tag{10}$$

The percent change β is also represented as the uncertainty factor, and it designates the extent to which an uncertain parameter changes. With the nominal value remaining constant, the bigger β is, the wider range the interval parameter varies within.

The perturbation method is convenient for implementation and is widely used in uncertainty fields. However, as the percent change β increases, the difference between the results obtained using the perturbation method and the exact solutions becomes significant. The bigger β is, the bigger the errors are. When dealing with uncertainty problems with large percent changes, the results may be wildly inaccurate. This is a prominent problem for the application of the perturbation method. To obtain the interval natural frequencies with a high level of accuracy, we employ a BP neural network.

3. MECHANISM OF A BP NEURAL NETWORK

An artificial neural network can simulate the human thought process and holds a strong non-linear mapping ability. The BP neural network, namely the error back-propagation neural network, is the most maturely studied neural network and has been successfully employed in many research fields. In this study, it is used to predict the bounds of natural frequencies for structures with interval parameters. The BP neural network is a multilayer forward, one-way transmission network, and it consists of three layers: input layer, hidden layer, and output layer. Firstly, the known parameters are inputted to the input layer. Subsequently, each neuron in the hidden layer produces an activation signal to the output layer. Finally, the neurons in the output layer produce a result based on the linear combination of the activations passed from the hidden layer. The topology structure of a BP neural network is presented in Figure 1.

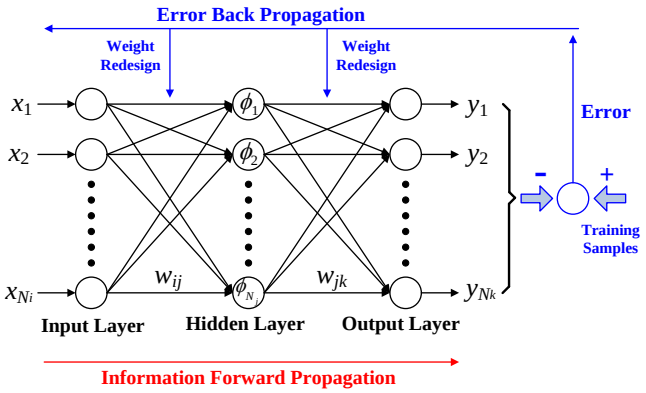


Figure 1. Topology structure of a BP neural network.

After training, the BP neural network can establish the correlation between the input data and output data, expressed as

$$f: \ \boldsymbol{x} \rightarrow \boldsymbol{y} \tag{11}$$

where f is the mathematical relationship function, $\boldsymbol{x} = [x_1, x_2, ..., x_{N_i}]$ is the N_i-dimensional input vector, and $\boldsymbol{y} = [y_1, y_2, ..., y_{N_k}]$ is the N_k-dimensional output vector. In between the input layer together with the output layer, there is a hidden layer. The hidden layer consists of multiple nodes, which are also called neurons.

The input layer firstly transfers the input data to the hidden layer. Each neuron in the hidden layer implements an excitation function. The excitation function in the neuron of the hidden layer usually uses a sigmoid function, namely:

$$\phi_j(u) = 1/(1+e^{-u}), \ \ (j = 1, 2, ..., N_j) \tag{12}$$

where N_j denotes the number of neurons of the hidden layer.

The number of neurons in the hidden layer affects the performance of BP neural networks. However, so far, there is no general rule to determine this number. The number of neurons in the hidden layer can be determined using the following relationship:

$$N_j = \log_2 N_i \tag{13}$$

where N_i represents the number of input variables. In most situations, the number of neurons in the hidden layer obtained using Eq. (13) can satisfy the accuracy requirement. If the accuracy requirement is not satisfied, we may increase the number of neurons in the hidden layer.

The input of the j^{th} neuron in the hidden layer is a_j, expressed as:

$$a_j = \sum_{i=1}^{N_i} (w_{ij} x_i - \theta_j), \ (i = 1, 2, ..., N_i, \ \ j = 1, 2, ..., N_j) \tag{14}$$

where w_{ij} is the weighting factor connecting the i^{th} input layer neuron and the j^{th} hidden layer neuron, and θ_j is the threshold value of the j^{th} hidden layer neuron.

The output of the j^{th} neuron in the hidden layer is b_j, expressed as:

$$b_j = \phi_j(a_j), \ (j = 1, 2, ..., N_j) \tag{15}$$

where ϕ_j is the excitation function, as given in Eq. (12).

Now calculate the output of the k^{th} neuron in the output layer, expressed as:

$$y_k = \sum_{j=1}^{N_j} (w_{jk} b_j - \theta_k),\ (j = 1, 2, ..., N_j,\ k = 1, 2, ..., N_k) \tag{16}$$

where w_{jk} is the weight corresponding to the connection between the j^{th} hidden layer neuron and the k^{th} output layer neuron, and θ_k is the threshold value of the k^{th} output layer neuron. The weights and threshold values reflect the contribution of a certain hidden layer neuron to an individual output.

A BP neural network can establish a certain mapping function from an N_i-dimensional space to an N_k-dimensional space. A set of input data, for which the corresponding outputs are already known, should be provided. These data are called the training samples. The weights and threshold values of the network can be determined after the training process. The training process of the network is essentially a minimization process of the error function. When a sample, indicated by s, is inputted to the BP neural network, an output vector can be calculated. The error function corresponding to this sample is the sum of the squared error of each output in the output layer, expressed as:

$$E^{(s)} = \frac{1}{2} \sum_{k=1}^{N_k} (y_k^{(s)} - d_k^{(s)})^2 \tag{17}$$

where $\boldsymbol{y}^{(s)} = [y_1^{(s)}, y_2^{(s)}, ..., y_q^{(s)}]$ is the network output vector corresponding to the s^{th} sample, and $\boldsymbol{d}^{(s)} = [d_1^{(s)}, d_2^{(s)}, ..., d_q^{(s)}]$ is the expected output vector corresponding to the s^{th} sample.

Suppose that there are N_s training samples in total. After all the samples are inputted to the BP neural network, the network error function can be obtained as follows:

$$E_{network} = \sum_{s=1}^{N_s} E^{(s)} = \frac{1}{2} \sum_{s=1}^{N_s} \sum_{k=1}^{N_k} (y_k^{(s)} - d_k^{(s)})^2 \tag{18}$$

where $E_{network}$ is called the network error function.

The values of the weights and thresholds are first randomly assigned, and then optimized using iteration procedures. This approach guarantees the robustness of the network for arbitrary inputs. The error corresponding to the k^{th} output at the t^{th} iteration step is defined as $e_k^{(t)}$, expressed as:

$$e_k^{(t)} = d_k - y_k^{(t)},\ \ (k = 1, 2, ..., N_k) \tag{19}$$

where d_k and $y_k^{(t)}$ are the desired output and network output for the k^{th} neuron in the output layer, respectively; the superscript (t) represents the t^{th} iteration step.

The iteration equations for the weights which connect two adjacent layers at the $(t+1)^{th}$ iteration step are expressed as follows:

$$\begin{cases} w_{ij}^{(t+1)} = w_{ij}^{(t)} + \eta b_j^{(t)}(1-b_j^{(t)})x_i \cdot (\sum_{k=1}^{N_k} w_{jk}^{(t)} e_k^{(t)}) \\ w_{jk}^{(t+1)} = w_{jk}^{(t)} + \eta b_j^{(t)} e_k^{(t)} \end{cases} \tag{20}$$

where η is called the learning rate, and $\eta \in (0,1)$; $i = 1,2,...,N_i$; $j = 1,2,...,N_j$; $k = 1,2,...,N_k$.

The iteration equations for the threshold values in the hidden layer and the output layer at the $(t+1)^{th}$ iteration step are expressed as follows:

$$\begin{cases} \theta_j^{(t+1)} = \theta_j^{(t)} + \eta b_j^{(t)}(1-b_j^{(t)}) \cdot (\sum_{k=1}^{N_k} w_{jk}^{(t)} e_k^{(t)}) \\ \theta_k^{(t+1)} = \theta_k^{(t)} + e_k^{(t)} \end{cases} \tag{21}$$

where η is also the learning rate; $j = 1,2,...,N_j$ and $k = 1,2,...,N_k$.

When the network error function $E_{network}$ in Eq. (18) is smaller than a specified small quantity e_0, (for example, $e_0 = 10^{-4}$), one can assume that the iterations for the network training have terminated. At this stage, the values of the weights and thresholds are assumed to have converged. The termination condition can be expressed as:

$$E_{network} < e_0 \tag{22}$$

When the termination condition is satisfied, the training process is completed. After the training process, the BP neural network is assumed to have successfully established the correct functional relationship between the inputs and the outputs. At this stage, the parameters of this network are determined, and the network can be used for many applications, such as functional approximation, image processing and pattern recognition. In this study, the BP neural network is adopted to evaluate the bounds of natural frequencies for structures with interval parameters.

4. BP NEURAL NETWORK-BASED METHOD FOR PREDICTION ON INTERVAL NATURAL FREQUENCIES

In this study, we proposed a BP neural network-based method to calculate the lower and upper bounds of natural frequencies for structures with interval parameters. The procedures of implementing this method are described in the subsequent sections.

4.1. Establishment of FEM Model

The finite element method (FEM) is a well-accepted computational tool for assessing the mechanical properties of structures. In this study, we use a FEM model to obtain the natural frequencies of a structure. The stiffness matrix $\boldsymbol{K}$ and mass matrix $\boldsymbol{M}$ can be obtained using the FEM model, and Eq. (1) can be adopted to calculate the natural frequencies. Usually, the lower order natural frequencies have greater effects on the dynamic behaviors than the higher order ones, and our study focuses on the lower order natural frequencies.

4.2. Designation of Inputs and Outputs

The purpose of this paper is to reveal the inherent law between uncertain parameters and natural frequencies. Owing to variances of the environment, measurement errors, or manufacturing inaccuracy, the structural parameters will definitely exhibit some uncertainties. The uncertain parameters are listed as a vector as follows:

$$\boldsymbol{a} = [a_1, a_2, ..., a_{N_a}]^T \tag{23}$$

where N_a represents the number of uncertain parameters. The uncertain parameters can be the Young's modulus, the mass density, or Poisson's ratio. Based on the interval analysis, the vector $\boldsymbol{a}$ is assumed to vary within an interval vector $\boldsymbol{a}^I$, i.e.,

$$\boldsymbol{a} \in \boldsymbol{a}^I = [a_1^I, a_2^I, ..., a_{N_a}^I]^T \tag{24}$$

or in the element form as follows:

$$a_l \in a_l^I = [\underline{a}_l, \overline{a}_l], \ (l = 1, 2, ..., N_a) \tag{25}$$

where a_l^I is an interval number, and $\underline{a}_l$, $\overline{a}_l$ denote the lower and upper bounds of a_l^I.

The uncertain parameters are designated as the inputs of the BP neural network, expressed as:

$$\boldsymbol{x} = \boldsymbol{a} \tag{26}$$

where $\boldsymbol{x}$ is the input vector expressed in Eq. (11). In addition, we have:

$$N_i = N_a \tag{27}$$

where N_i is the neuron amount in the input layer of a BP neural network.

The natural frequencies of concern are listed as a vector as follows:

$$\boldsymbol{f} = [f_1, f_2, ..., f_{N_f}]^T \tag{28}$$

where N_f is the order of frequencies we care about. In our following numerical example, N_f = 4, i.e., our study focuses on the first 4 natural frequencies.

The natural frequencies of concern are designated as the inputs of the BP neural network, expressed as:

$$\boldsymbol{y} = \boldsymbol{f} \tag{29}$$

where $\boldsymbol{y}$ is the output vector expressed in Eq. (11). In addition, we have:

$$N_k = N_f \tag{30}$$

where N_k is the number of neurons in the output layer of a BP neural network.

4.3. Generation of Training Samples

Neural networks are mathematical models which are strictly data-driven. The prediction accuracy of a BP neural network is highly dependent on the training samples, which are designated to train the network to reveal the correct mathematical relationship between the inputs and outputs. Thus, the selection of training samples plays a significant role in the BP neural network establishment. The training samples are required to cover the design variable space uniformly. Usually, samples are generated using the design of experiments (DOE), such as orthogonal arrays design, full factorial design, Latin hypercube design, and optimal Latin hypercube design. In this study, the training samples are generated by an optimal Latin hypercube design, which is an improvement over the classical Latin hypercube design. The optimal Latin hypercube design distributes the sample data over the design variable space in a uniform manner and can avoid the occurrence of data clustering.

There is no rule for determining the appropriate number of training samples for a BP neural network. It is preferable to analyse as many samples as possible to obtain an accurate network. However, too many samples mean a heavy computational load, and often, it is unnecessary. Therefore, the number of samples should be determined by balancing accuracy and efficiency. The number of training samples is firstly assigned with an initial value N_{s0}, then if this number is not enough, we may enhance the accuracy by applying an increment ΔN_s . The numbers of different sets of training samples are expressed as follows:

$$N_1 = N_{s_0} + \Delta N_s,\ N_2 = N_{s_0} + 2\times\Delta N_s, \cdots\cdots, N_s = N_{s_0} + s\times\Delta N_s \tag{31}$$

where N_s represents the number of the s^{th} set of training samples. In the following sub-section 4.7, the termination condition for N_s is proposed. When the termination condition is satisfied, the number of training samples can be assumed to be large enough.

The s^{th} set of training samples can be expressed as follows

$$S_{input}^{(N_s)} = \{(\boldsymbol{a}^{(1)}, \boldsymbol{a}^{(2)}, ..., \boldsymbol{a}^{(i)}, ..., \boldsymbol{a}^{(N_s)}) \mid \boldsymbol{a}^{(i)} \in \boldsymbol{a}^{I}, 1 \le i \le N_s\} \tag{32}$$

where $S_{input}^{(N_s)}$ represents the s^{th} set of training samples; $\boldsymbol{a}^{(i)}$ is the i^{th} input vector and varies within the interval $\boldsymbol{a}^{I}$; the samples $\boldsymbol{a}^{(i)}$ $(1 \le i \le N_s)$ are distributed by the optimal Latin hypercube design; N_s is the number of training samples.

The desired outputs corresponding to the training samples should be obtained using the FEM analysis, expressed as:

$$S_{output}^{(N_s)} = \{\boldsymbol{f}^{(1)}, \boldsymbol{f}^{(2)}, ... \boldsymbol{f}^{(i)} ..., \boldsymbol{f}^{(N_s)} \mid \boldsymbol{f}^{(i)} = [f_1^{(i)}, f_2^{(i)}, ..., f_{N_f}^{(i)}], 1 \le i \le N_s\} \tag{33}$$

where $S_{output}^{(N_s)}$ represents the desired outputs corresponding to N_s training samples; $\boldsymbol{f}^{(i)}$ is the i^{th} output natural frequency vector; N_f is the order of frequencies we care about.

4.4. Establishment of BP Neural Network

The training samples, presented in Eq. (32), and their desired outputs, presented in Eq. (33), are employed to create the BP neural network. The correctness of the outputs is based on the topology of the network. In Section 3, the procedures of how to determine the values of weights and thresholds are presented, and the process of determining these parameters is called the training process. After the BP neural network is successfully trained, the mapping function between the input data and the output data can be obtained. At this stage, the BP neural network corresponding to N_s training samples is established, and we can retrieve the relationship between the uncertain parameters and the natural frequencies. To a certain extent, this implicit relationship becomes explicit by utilizing the BP neural network. Subsequently, the established BP neural network can be employed to predict the lower and upper bounds of natural frequencies for structures with interval parameters.

4.5. Generation of Testing Samples

The established BP neural network reveals the inherent law between the uncertain parameters and the natural frequencies. However, our direction is to calculate the lower and upper bounds of natural frequencies. To achieve this goal, we need to generate testing samples. The purpose of generating testing samples is to set them as inputs to the BP neural network, and we observe the maximum and minimum from the corresponding outputs. As for the sampling distribution, we also apply the technique of optimal Latin hypercube design, which distributes the sample data over the design variable space uniformly and avoids the data clustering problem. The number of testing samples is much larger than that of the training samples, because we want these samples to fill the design variable space as thoroughly as possible.

The set of testing samples can be expressed as follows:

$$T_{input} = \{(\boldsymbol{a}^{(1)}, \boldsymbol{a}^{(2)}, ..., \boldsymbol{a}^{(i)}, ..., \boldsymbol{a}^{(N_T)}) \mid \boldsymbol{a}^{(i)} \in \boldsymbol{a}^{I}, 1 \le i \le N_T\} \tag{34}$$

where T_{input} represents the set of testing samples; $\boldsymbol{a}^{(i)}$ is the i^{th} input vector and varies within the interval $\boldsymbol{a}^{I}$; the samples $\boldsymbol{a}^{(i)}$ $(1 \le i \le N_T)$ are distributed by the optimal Latin hypercube design; N_T denotes the number of testing samples and in our study, $N_T = 10^6$.

4.6. Calculation of Lower and Upper Bounds

At this stage, the BP neural network corresponding to Ns training samples has been established, and the testing samples in Eq. (34) can be inputted into the network. The outputs corresponding to these testing samples are called the testing outputs, expressed as:

$$T_{output}^{(N_s)} = \{\boldsymbol{f}^{(1)}, \boldsymbol{f}^{(2)}, ... \boldsymbol{f}^{(i)}, ..., \boldsymbol{f}^{(N_T)} \mid \boldsymbol{f}^{(i)} = [f_1^{(i)}, f_2^{(i)}, ..., f_{N_f}^{(i)}], 1 \le i \le N_T\} \tag{35}$$

where $T_{output}^{(N_s)}$ represents the set of testing outputs corresponding to N_s training samples; $\boldsymbol{f}^{(i)}$ is the i^{th} output natural frequency vector; N_f is the order of frequencies we care about.

The upper bound vector of the natural frequencies is the maximum of the testing output set $T_{output}^{(N_s)}$, expressed as:

$$\overline{\boldsymbol{f}}^{(N_s)} = \max\{\boldsymbol{f}^{(i)} \mid \boldsymbol{f}^{(i)} \in T_{output}^{(N_s)}, 1 \le i \le \mathrm{N}_T\} \tag{36}$$

or in the element form expressed as:

$$\overline{\boldsymbol{f}}^{(N_s)} = [\overline{f}_1^{(N_s)}, \overline{f}_2^{(N_s)}, ..., \overline{f}_{N_f}^{(N_s)}] \tag{37}$$

where $\overline{\boldsymbol{f}}^{(N_s)}$ is the upper bound vector corresponding to N_s training samples.

Similarly, the lower bound vector of the natural frequencies is the minimum of the testing output set $T_{output}^{(N_s)}$, expressed as:

$$\underline{\boldsymbol{f}}^{(N_s)} = \min\{\boldsymbol{f}^{(i)} \mid \boldsymbol{f}^{(i)} \in T_{output}^{(N_s)}, 1 \le i \le \mathrm{N}_T\} \tag{38}$$

or in the element form expressed as:

$$\underline{\boldsymbol{f}}^{(N_s)} = [\underline{f}_1^{(N_s)}, \underline{f}_2^{(N_s)}, ..., \underline{f}_{N_f}^{(N_s)}] \tag{39}$$

where $\underline{\boldsymbol{f}}^{(N_s)}$ is the upper bound vector corresponding to N_s training samples.

Now, the lower and upper bounds of natural frequencies are obtained using the established BP neural network, and presented in Eq. (36) and Eq. (38). It should be noted that the bound results obtained here correspond to Ns training samples, and if the bound results do not satisfy the termination condition in sub-section 4.7, the number of training samples should be increased. Besides, the lower and upper bounds should be calculated again with the increased training samples.

4.7. Termination Condition

The number of training samples is of tremendous significance for the BP neural network establishment, because insufficient training samples may lead to inaccurate outputs, whereas, excessive training samples consume too many computing resources. It is crucial to determine a proper number of training samples. In this study, the number of training samples is presented in Eq. (31), and we put forward the termination condition below to check whether the number is enough or not. This is a feasible approach to strike a good balance between the demands of accuracy and efficiency for determining the number of training samples.

The lower and upper bounds of natural frequencies corresponding to N_s training samples are $\underline{\boldsymbol{f}}^{(N_s)}$ and $\overline{\boldsymbol{f}}^{(N_s)}$, respectively, and the lower and upper bounds of natural frequencies corresponding to $N_{s\text{-}1}$ training samples are $\underline{\boldsymbol{f}}^{(N_{s-1})}$ and $\overline{\boldsymbol{f}}^{(N_{s-1})}$, respectively. We propose a bound error function based on the adjacent sets of bounds obtained using different training samples, expressed as:

$$E_{bound} = \| \underline{\boldsymbol{f}}^{(N_s)} - \underline{\boldsymbol{f}}^{(N_{s-1})} \|^2 + \| \overline{\boldsymbol{f}}^{(N_s)} - \overline{\boldsymbol{f}}^{(N_{s-1})} \|^2 \tag{40}$$

where E_{bound} is the bound error function; the symbol || || is the Euclidean norm of a vector; N_s and N_{s-1} are the numbers of different training samples, presented in Eq. (31).

The bound error function E_{bound} can also be expressed in the element form as follows:

$$\begin{aligned} E_{bound} = & [\underline{f}_1^{(N_s)} - \underline{f}_1^{(N_{s-1})}]^2 + [\underline{f}_2^{(N_s)} - \underline{f}_2^{(N_{s-1})}]^2 + ... + [\underline{f}_{N_f}^{(N_s)} - \underline{f}_{N_f}^{(N_{s-1})}]^2 \\ & + [\overline{f}_1^{(N_s)} - \overline{f}_1^{(N_{s-1})}]^2 + [\overline{f}_2^{(N_s)} - \overline{f}_2^{(N_{s-1})}]^2 + ... + [\overline{f}_{N_f}^{(N_s)} - \overline{f}_{N_f}^{(N_{s-1})}]^2 \end{aligned} \tag{41}$$

where N_f is the order of natural frequencies of concern.

When the bound error function E_{bound} in Eq. (40) is smaller than a specified small quantity ε_0, (for example, $\varepsilon_0 = 0.1$), we can assume that the value N_s is adequate for network establishment, and we do not have to increase the number of training samples anymore. This termination condition can be expressed as follows:

$$E_{bound} < \varepsilon \tag{42}$$

and at this stage, the calculation of lower and upper bounds of natural frequencies is assumed to be completed.

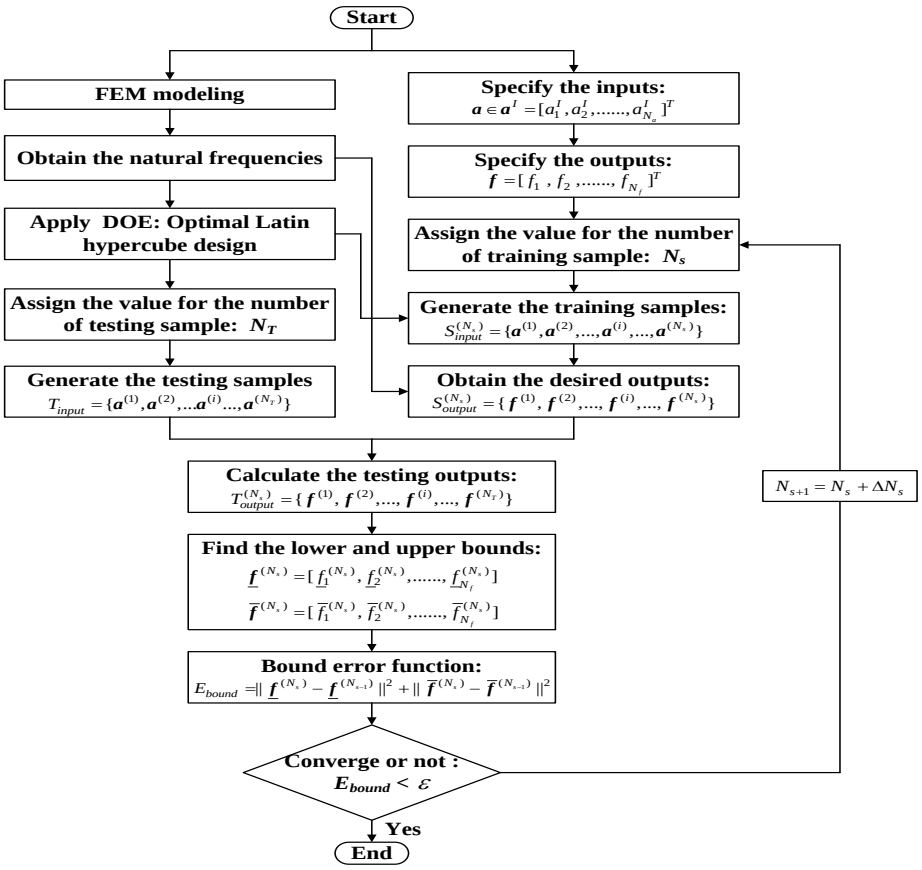


Figure 2. Flowchart of the proposed BP neural network-based method.

If the termination condition is not satisfied, the number of training samples should be increased, expressed as:

$$N_{s+1} = N_s + \Delta N_s \tag{43}$$

where N_{s+1} correspond to the $(s+1)^{th}$ set of training samples, and the BP neural network should be re-established. In addition, the lower and upper bounds of natural frequencies should be re-calculated.

Now, the BP neural network based method to predict the lower and upper bounds of natural frequencies for structures with uncertain-but-bounded parameters has been thoroughly put forward, and in the next section, a numerical example is tested to demonstrate the feasibility of the proposed method. To illustrate the procedures of the proposed method, a flowchart is presented in Figure 2.

5. NUMERICAL EXAMPLE

In order to demonstrate the feasibility of the proposed method to predict the interval of natural frequencies of structures with interval parameters, the following numerical example is presented. The accuracy of the proposed method is compared with that of the perturbation method. The lower and upper bounds calculated using the Monte Carlo simulation are regarded as reference solutions. The Monte Carlo simulation obtains the distribution of responses through an extremely large number of samples and is very time-consuming. Thus, the Monte Carlo simulation is always used to verify the accuracy of other methods; however, it is not convenient or suitable for engineering applications.

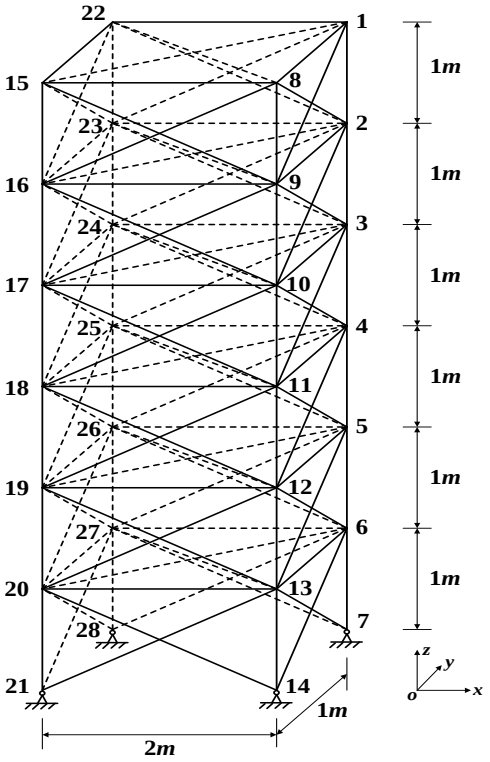


Figure 3. A 108-bar space truss structure

Consider a 108-bar space truss structure. Figure 3 indicates the dimensions and boundary conditions of the truss. The cross-sectional area of each rod is $A = 1.0 \times 10^{-4}$ m^2. Young's modulus of the material is $E = 210$ GPa, the mass density is $\rho = 7900$ kg/m^3, and Poisson's ratio is $\nu = 0.3$. In FEM analysis, each bar is regarded as a rod element. So, there are 108 elements in total. When the structural parameters are assigned their nominal values, we can obtain the nominal natural frequencies by using the existing FEM theory. The first 4 nominal natural frequencies are listed in Table 1.

Table 1. First 4 nominal natural frequencies (Unit: Hz)

Order	f_1	f_2	f_3	f_4
Value	13.6241	25.6548	49.4426	69.4558

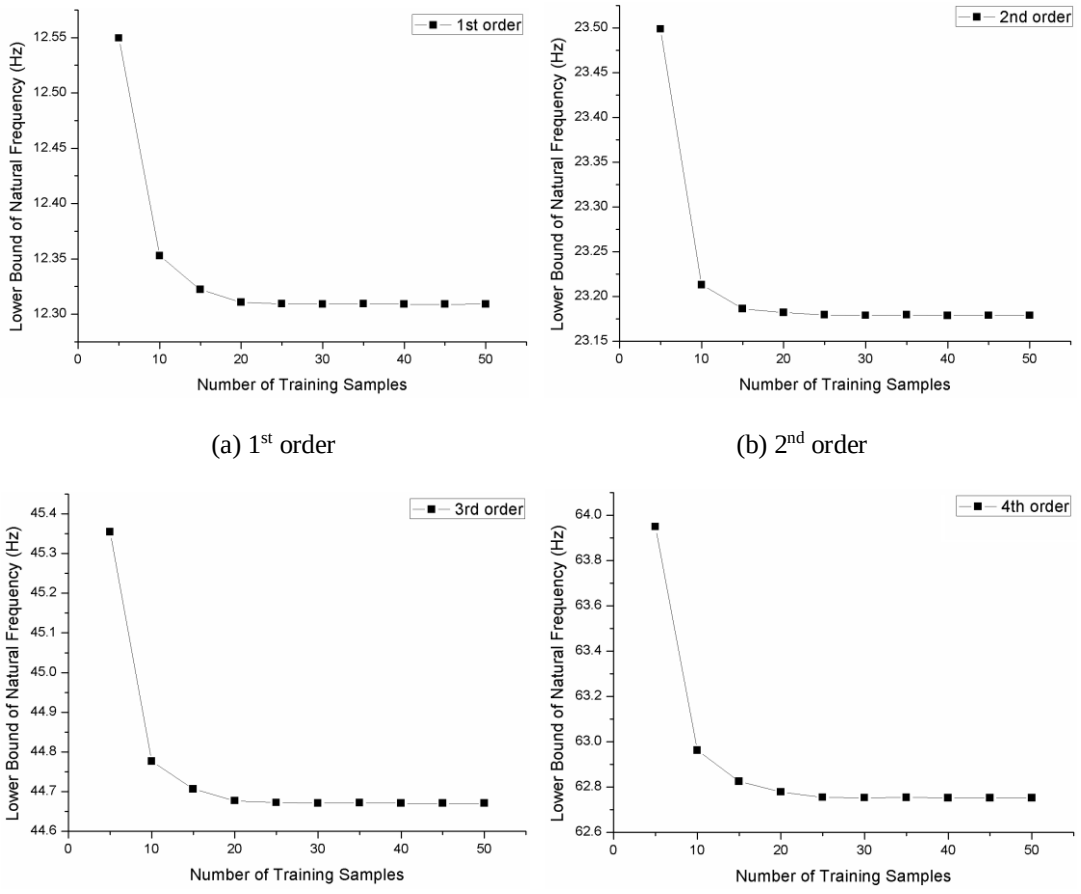
Owing to variances of the environment, manufacturing inaccuracy, or measurement errors, the parameters exhibit some uncertainties. In this example, Young's modulus and the mass density are considered to be the uncertain-but-bounded parameters. Young's modulus is assumed to vary within the interval $E^I = [1-\beta,\ 1+\beta] \cdot E^c$, where $E^c = 210$ GPa; the mass density is assumed to vary within the interval $\rho^I = [1-\beta,\ 1+\beta] \cdot \rho^c$, where $\rho^c = 7900$ kg/m^3; β is the percent change or also called the uncertainty factor, and in this example it is assigned the following values: β = 2%, 4%, 6%, 8%, 10%.

We employ the proposed method, the perturbation method and the Monte Carlo simulation to calculate the lower and upper bounds of the first 4 natural frequencies. The training samples play a significant role in neural network establishment. In this example, the sample data for training are generated by the optimal Latin hypercube design, which distributes the sample data over the design variable space in a uniform manner and can avoid the occurrence of data clustering. To study the effect of the sample number on BP neural network, we use different numbers of samples for training.

When $\beta = 10\%$, the lower bounds of natural frequencies obtained using the BP neural network corresponding to different numbers of training samples are listed in Table 2 and plotted in Figure 4; the upper bounds of natural frequencies obtained using the BP neural network corresponding to different numbers of training samples are listed in Table 3 and plotted in Figure 5. It can be observed that when the number of samples is greater than 40, the difference between the bound results of the two adjacent sample sets is smaller than 0.001. At this stage, the results can be assumed to have converged.

Table 2. Lower bounds of natural frequencies (Unit: Hz) obtained using the BP neural network with respect to different numbers of training samples (β=10%)

Number of Training Samples	$\underline{f}_1$	$\underline{f}_2$	$\underline{f}_3$	$\underline{f}_4$
5	12.5496	23.4988	45.3548	63.9491
10	12.3528	23.2130	44.7763	62.9616
15	12.3223	23.1863	44.7066	62.8249
20	12.3109	23.1822	44.6772	62.7784
25	12.3095	23.1795	44.6721	62.7543
30	12.3092	23.1789	44.6709	62.7527
35	12.3094	23.1794	44.6718	62.7540
40	12.3091	23.1787	44.6706	62.7522
45	12.3091	23.1788	44.6707	62.7524
50	12.3091	23.1788	44.6707	62.7524



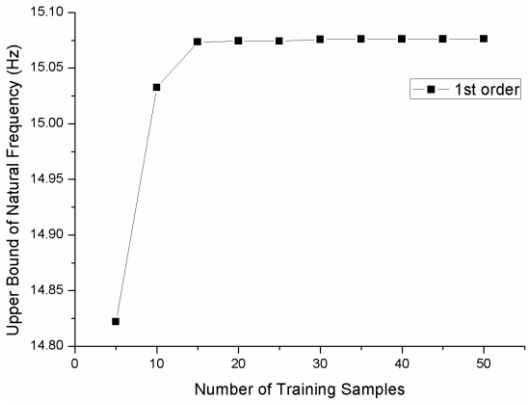
(a) 1st order (b) 2nd order

(c) 3rd order (d) 4th order

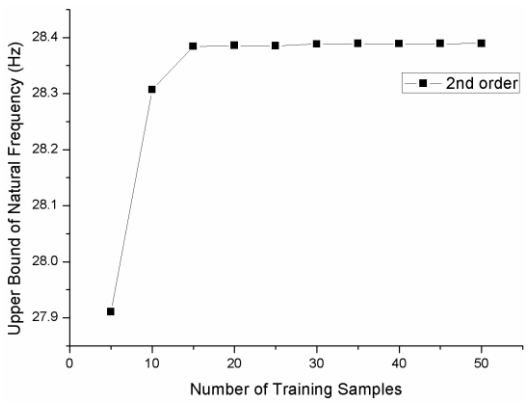
Figure 4. Lower bounds of natural frequencies (Unit: Hz) obtained using the BP neural network with respect to different numbers of training samples (β=10%)

Table 3. Upper bounds of natural frequencies (Unit: Hz) obtained using the BP neural network with respect to different numbers of training samples (β=10%)

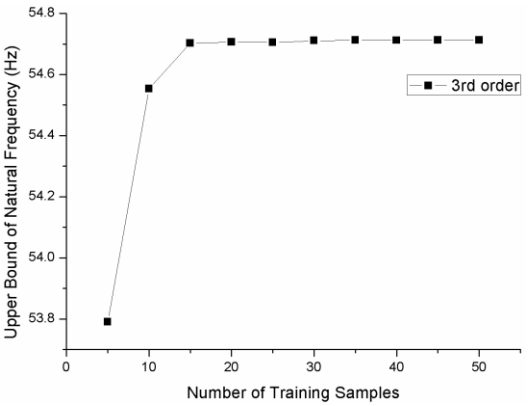
Number of Training Samples	$\overline{f}_1$	$\overline{f}_2$	$\overline{f}_3$	$\overline{f}_4$
5	14.8221	27.9108	53.7903	75.5634
10	15.0326	28.3070	54.5539	76.6361
15	15.0736	28.3843	54.7028	76.8453
20	15.0744	28.3859	54.7059	76.8496
25	15.0742	28.3854	54.7050	76.8483
30	15.0758	28.3884	54.7107	76.8563
35	15.0762	28.3892	54.7122	76.8585
40	15.0761	28.3890	54.7119	76.8580
45	15.0762	28.3892	54.7122	76.8585
50	15.0763	28.3894	54.7127	76.8592



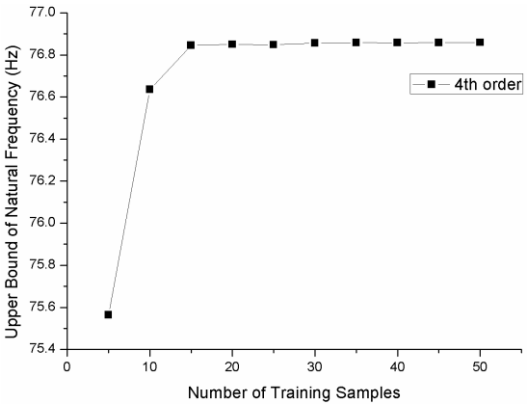
(a) 1st order



(b) 2nd order



(c) 3rd order



(d) 4th order

Figure 5. Upper bounds of natural frequencies (Unit: Hz) obtained using the BP neural network with respect to different numbers of training samples (β=10%)

The lower and upper bounds of the first 4 natural frequencies obtained using the proposed method, the perturbation method and the Monte Carlo simulation with respect to different values of percent change β are listed in Tables 4 to 7 and plotted in Figure 6.

Table 4. Lower and upper bounds of the 1st natural frequency with respect to different values of percent change β

β	$\underline{f}_1$			$\overline{f}_1$		
	BP Neural Network	Perturbation Method	Monte Carlo Simulation	BP Neural Network	Perturbation Method	Monte Carlo Simulation
0	13.6241	13.6241	13.6241	13.6241	13.6241	13.6241
0.02	13.3517	13.3497	13.3543	13.9028	13.9048	13.8993
0.04	13.0837	13.0658	13.0896	14.1852	14.2022	14.1804
0.06	12.8211	12.7788	12.8298	14.4762	14.5168	14.4676
0.08	12.5601	12.4750	12.5745	14.7756	14.8491	14.7613
0.10	12.3091	12.1782	12.3234	15.0763	15.1995	15.0620

Table 5. Lower and upper bounds of the 2nd natural frequency with respect to different values of percent change β

β	$\underline{f}_2$			$\overline{f}_2$		
	BP Neural Network	Perturbation Method	Monte Carlo Simulation	BP Neural Network	Perturbation Method	Monte Carlo Simulation
0	25.6548	25.6548	25.6548	25.6548	25.6548	25.6548
0.02	25.1419	25.1365	25.1468	26.1797	26.1834	26.1732
0.04	24.6372	24.6073	24.6484	26.7099	26.7435	26.7024
0.06	24.1429	24.0664	24.1591	27.2664	27.3359	27.2432
0.08	23.6513	23.5131	23.6784	27.8234	27.9615	27.7963
0.10	23.1788	22.9467	23.2057	28.3894	28.6215	28.3625

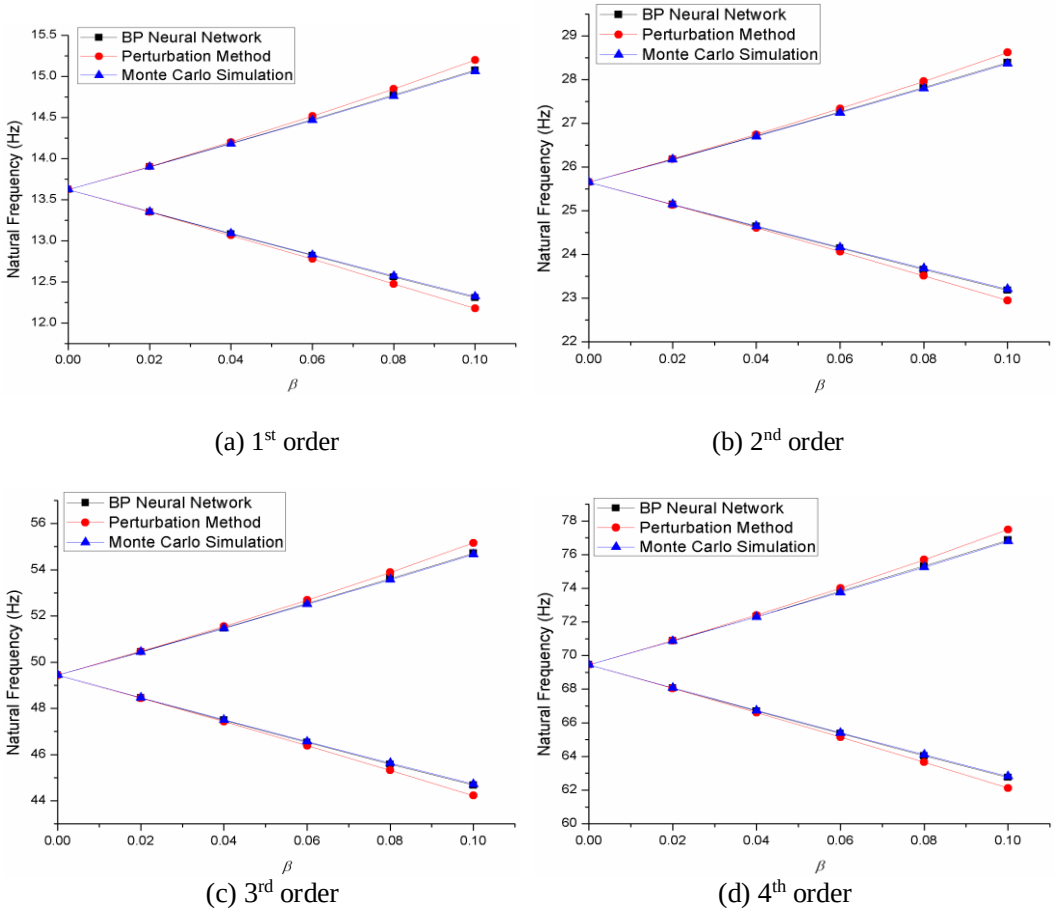
Table 6. Lower and upper bounds of the 3rd natural frequency with respect to different values of percent change β

β	$\underline{f}_3$			$\overline{f}_3$		
	BP Neural Network	Perturbation Method	Monte Carlo Simulation	BP Neural Network	Perturbation Method	Monte Carlo Simulation
0	49.4426	49.4426	49.4426	49.4426	49.4426	49.4426
0.02	48.4540	48.4436	48.4634	50.4540	50.4613	50.4415
0.04	47.4814	47.4236	47.5029	51.4759	51.5407	51.4614
0.06	46.5286	46.3813	46.5599	52.5449	52.6823	52.5037
0.08	45.5813	45.3150	45.6334	53.6218	53.8881	53.5697
0.10	44.6707	44.2233	44.7225	54.7127	55.1601	54.6608

Table 7. Lower and upper bounds of the 4th natural frequency with respect to different values of percent change β

β	$\underline{f}_4$			$\overline{f}_4$		
	BP Neural Network	Perturbation Method	Monte Carlo Simulation	BP Neural Network	Perturbation Method	Monte Carlo Simulation
0	69.4558	69.4558	69.4558	69.4558	69.4558	69.4558
0.02	68.0671	68.0525	68.0803	70.8766	70.8869	70.8591
0.04	66.7007	66.6197	66.7310	72.3122	72.4032	72.2919
0.06	65.3624	65.1554	65.4063	73.8199	74.0069	73.7560
0.08	64.0316	63.6575	64.1048	75.3267	75.7007	75.2534
0.10	62.7524	62.1239	62.8251	76.8592	77.4875	76.7863

The results obtained using the Monte Carlo simulation are referred to as exact solutions. The sampling size of the Monte Carlo simulation is 10^6. It can be observed that when the percent change β is relatively small, there is no fundamental difference between the bounds obtained using the proposed method and the perturbation method. However, as the percent change β increases, the difference between the bounds obtained using the proposed method and the perturbation method is significant. The comparison indicates that the bounds obtained using the proposed method are more accurate than those obtained using the perturbation method.



(a) 1st order (b) 2nd order

(c) 3rd order (d) 4th order

Figure 6. Bounds of the first 4 natural frequencies with respect to different values of percent change β

Table 8. Details of the computation environment

Computation Environment	
CPU :	3.40 GHz
Memory :	16 GB
System Type :	64 bit
Number of Processors :	8
Operating System :	Windows 7
Programing Environment :	ANSYS 16.0 APDL; MATLAB R2013a

The computation time of the proposed method, the perturbation method and Monte Carlo simulation are presented. The details of the computation environment for this numerical example are listed in Table 8. When the percent change β=10%, the computation time corresponding to different methods are listed in Table 9. The Monte Carlo simulation requires a large number of FEM computation samplings, which leads to a heavy computation load. Especially when the structure is complicated, even a one-time FEM computation would be time-consuming. The proposed method also requires FEM computation samplings, but the sampling size is much smaller than that of the Monte Carlo simulation. Thus, the proposed method is more applicable for engineering structures than the Monte Carlo simulation.

Table 9. Comparison of the computation time corresponding to different methods in the numerical example when β =10%

Method	BP Neural Network	Perturbation Method	Monte Carlo Simulation
Computation Time	145.39 s	32.12 s	7.79×10^4 s

The computation time of our proposed method is more than that of the perturbation method within an acceptable range, in this example, it is about 113 s longer, but the comparison results in Figure 6 indicate that the accuracy of the proposed method is much higher than that of the perturbation method. The results obtained using the proposed method are in good accordance with those obtained using the Monte Carlo simulation. The proposed method can strike a good balance between the demands of accuracy and efficiency. This is the principal advantage of the proposed method to predict the interval natural frequencies of structures with interval parameters.

6. CONCLUSION

In this study, a BP neural network-based method was proposed to predict the interval natural frequencies of structures with uncertain-but-bounded parameters. The inherent law between the uncertain parameters and the natural frequencies is revealed using a BP neural network. A numerical example is employed to manifest the feasibility of the proposed method. The results indicate that the interval natural frequencies obtained using the proposed method are of higher accuracy than those obtained using the perturbation method, especially when the varying ranges of interval parameters are large. Our purpose is not only to put forward a novel method to predict the interval natural frequencies with a high level of accuracy but also to manifest the enormous potential of BP neural networks for solving uncertain problems in mechanics, such as uncertain static problems, uncertain dynamic problems, and uncertain buckling problems.

REFERENCES

[1] Y. Ben-Haim & I. Elishakoff, (1990) *Convex Models of Uncertainty in Applied Mechanics*, Elsevier, New York.

[2] X.J. Wang & L. Wang, (2011) "Uncertainty quantification and propagation analysis of structures based on the measurement data", *Mathematical and Computer Modelling*, Vol. 54, No. 11, pp2725–2735.

[3] Z. Lv & Z.P. Qiu, (2016) "A direct probabilistic approach to solve state equations for nonlinear systems under random excitation", *Acta Mechanica Sinica*, Vol. 32, No. 5, pp941–958.

[4] S.S. Rao & L. Berke, (1997) "Analysis of uncertain structural systems using interval analysis", *AIAA Journal*, 35, No. 4, pp727–735.

[5] Z.P. Qiu & L. Wang, (2016) "The need for introduction of non-probabilistic interval conceptions into structural analysis and design", *Science China: Physics, Mechanics & Astronomy*, Vol. 59, No. 11, pp114632.

[6] L. Wang, X.J. Wang & Y. Xia, (2014) "Hybrid reliability analysis of structures with multi-source uncertainties", *Acta Mechanica*. Vol. 225, No. 2, pp413–430.

[7] G. Augusti, A. Baratta & F. Casciati, (1984) *Probabilistic Methods in Structural Engineering*, CRC Press, Boca Raton.

[8] I. Elishakoff & J.T.P. Yao, (1983) *Probabilistic Methods in the Theory of Structures*, Wiley Press, New York.

[9] L. Wang, D.L. Liu & Z.P. Qiu, (2017) "A novel method of non-probabilistic reliability-based topology optimization corresponding to continuum structures with unknown but bounded uncertainties", *Computer Methods in Applied Mechanics and Engineering*, Vol. 326, pp573–595.

[10] L. Jaulin, M. Kieffer & O. Didrit, (2001) *Applied Interval Analysis*, Springer, Berlin.

[11] R.E. Moore, (1979) *Methods and Applications of Interval Analysis*, Prentice-Hall, London.

[12] Z.P. Qiu & Z. Lv, (2017) "The vertex solution theorem and its coupled framework for static analysis of structures with interval parameters", *International Journal for Numerical Methods in Engineering*, Vol. 112, No. 7, pp711–736.

[13] L. Wang & X.J. Wang, (2017) "A novel method of Newton iteration-based interval analysis for multidisciplinary systems", *Science China: Physics, Mechanics & Astronomy*, Vol. 60, No. 9, pp094611.

[14] W. Verhaeghe, W. Desmet, D. Vandepitte & D. Moens, (2013) "Interval fields to represent uncertainty on the output side of a static FE analysis", *Computer Methods in Applied Mechanics and Engineering*, Vol. 260, pp50–62.

[15] N. Impollonia & G. Muscolino, (2011) "Interval analysis of structures with uncertain-but-bounded axial stiffness", *Computer Methods in Applied Mechanics and Engineering*, Vol. 200, pp1945–1962.

[16] S.H. Chen & X.W. Yang, (2000) "Interval finite element method for beam structures", *Finite Elements in Analysis and Design*, Vol. 34, pp75–88.

[17] Z. Lv, Z.P. Qiu & Q. Li, (2017) "An interval reduced basis approach and its integrated framework for acoustic response analysis of coupled structural-acoustic system", *Journal of Computational Acoustics*, Vol. 25, No. 3, pp1750009.

[18] S.H. Chen, (2007) *Matrix Perturbation Theory in Structural Dynamic Design*, Science Press, Beijing.

[19] J.C. Chen & B.K. Wade, (1977) "Matrix perturbation for structural dynamics", *AIAA Journal*, Vol. 15, No. 8, pp1095–1100.

[20] C.S. Rudisill, (1974) "Derivatives of eigenvalues and eigenvectors for a general matrix", *AIAA Journal*, Vol. 12, No. 1, pp721–722.

[21] M. Gevrey, I. Dimopoulos & S. Lek, (2003) "Review and comparison of methods to study the contribution of variables in artificial neural network models", *Ecological Modelling*, Vol. 160, pp249–264.

[22] G.N. Xu & Q. Zhang, (2012) "Safety assessment of the general overhead traveling crane metal structure based on BP neural network", *Applied Mechanics and Materials*, Vol. 101, pp15–20.

[23] S.M. Yuan, Z.F. Zhan & Y. Li, (2013) "Optimum design of machine tool structures based on BP neural network and genetic algorithm", *Advanced Materials Research*, vol. 655, pp1291–1295.

[24] G.M. Yang, C.S. Gu, Y. Huang & K. Yang, (2014) "BP neural network integration model research for hydraulic metal structure health diagnosing", *International Journal of Computational Intelligence Systems*, Vol. 7, No. 6, pp1148–1158.

[25] Q. Li, J.Y. Yu, B.C. Mu & X.D. Sun, (2006) "BP neural network prediction of the mechanical properties of porous NiTi shape memory alloy prepared by thermal explosion reaction", *Materials Science and Engineering*, Vol. 419, pp214–217.

PREPERFORMANCE TESTING OF A WEBSITE

Sushma Suryadevara and Shahid Ali

Department of Information Technology, AGI Institute, Auckland, New Zealand

ABSTRACT

This study was conducted on the importance of performance testing of web applications and analyzing the bottleneck applications. This paper highlights performance testing based on load tests. Everyone wants the application to be very fast, at the same time, reliability of the application also plays an important role, such that user's satisfaction is the push for performance testing of a given application. Performance testing determines a few aspects of system performance under the pre-defined workload. In this study JMeter performance testing tool was used to implement and execute the test cases. The first load test was calculated with 200 users which was increased to 500 users and their throughput, median, average response time and deviation were calculated.

KEYWORDS

Performance testing, load balancing, threads, throughput, JMeter, load test

1. INTRODUCTION

Flipmind company started in 2005 by Michael Jones Owner and Director of the company with a dream helping clients. Flipmind creates software to help the business succeed in E-commerce Consulting & Development, Website Design, Mobile Development, and Performance Analysis & Optimization, etc. The company had delivered a lot of projects in New Zealand such as Forza, Park & Fly, Money Place, My Ride, etc. The company has worked with Torpedo7 since 2006 providing BA, systems architecture, web & e-commerce development & load testing services, etc. Flipmind built the eCommerce platform powering Torpedo7 from the ground up. In New Zealand and Australia, Torpedo7 sells thousands of products and you would be hard- pressed to find a cyclist that does not buy from them. Flipmind company started in 2005 by Michael Jones Owner and Director of the company with a dream helping clients. Flipmind creates software to help the business succeed in E-commerce Consulting & Development, Website Design, Mobile Development, and Performance Analysis & Optimization, etc. The company had delivered a lot of projects in New Zealand such as Forza, Park & Fly, Money Place, My Ride, etc. The company has worked with Torpedo7 since 2006 providing BA, systems architecture, web & e-commerce development & load testing services, etc. Flipmind built the eCommerce platform powering Torpedo7 from the ground up. In New Zealand and Australia, Torpedo7 sells thousands of products and you would be hard-pressed to find a cyclist that does not buy from them. Flipmind is one of the most rapidly growing company in New Zealand as it offers high-class service such as website development, testing and its maintenance updating the data regularly to its stakeholders. Currently working on a project named as Torpedo7 website. Torpedo7 is the most popular outdoor store in New Zealand. We can browse our huge selection of snow and water gear, bikes comfortable outdoor clothing for women, men, and kids, camping equipment, hiking gear, bags & packs of all sizes, trampolines, and outdoor technology use in your next adventure. Nowadays, for every web application, everyone wants everything to be fast and at the same time, there is concern about the reliability of usage. Most of the users want to load their webpage

David C. Wyld et al. (Eds): ISPR, ARIN, MLCL, NBIoT - 2020
pp. 33-52, 2020. CS & IT - CSCP 2020 DOI: 10.5121/csit.2020.100703

rapidly fast so that they can finish and move to another work. If a page took a long time to load, the users will end up the task. The economic growth of any organization depends on the web application. If the application is not running fast, people lost interest in that application. So, it is important to do the performance testing of a web application for any organization. The load testing is done to test the performance of a website. By doing this performance testing we can know the system behaviour while handling specific load given by the customer to the system [4].

The main challenge for this project was to conduct the is load test to torpedo7.co.nz website for Black Friday and Christmas to make sure the website does not crash on those days. Load testing before Black Friday and Christmas or on any big event is very important for companies. The performance testing helps to fix bottlenecks, errors, and bugs due to high traffic spikes on the website. The cost of failure is the direct loss of sales and money to retailers. Using load testing failure can be easily avoided.

The scope of my project is to do a load test of the Torpedo7 website using JMeter for the Homepage, Shop feature(Snow, Technology, Clothing, Water, Bike), Service feature (Bike workshop), Search functionality, adding the item to the cart and proceed to checkout with 200, 500 users hence compared and analysed the results using summary report, graphs, CSV file and HTML report. The core of this objective is to make sure that the current application is optimal for handling increased traffic on the website. Hence to analyze and measure the performance of the Torepdo7 website we will be using the JMeter tool.

This research is organized as follow: Section 2 focuses on the literature review of various studies concentrating on automation regression. Section 3 is focused on the research methodology for this research. Section 4 of this research is focused on research execution results. Discussion to results of this research are provided in section 5. In section 6 conclusion to the research is provided. Finally, section 7 is dedicated towards the future work recommendations.

2. LITERATURE REVIEW

Many studies have been conducted in the past in favour of performance and load testing, we will investigate those studies, respectively.

A research was conducted on the Importance of performance testing of web applications and analysing the bottleneck applications [4]. This study highlights the performance testing based on the load test. Everyone wants the application to be very fast, at the same time, reliability of the application also plays an important role, such that user's satisfaction is the push for performance testing of a given application. Performance testing determines a few aspects of system performance under the pre-defined workload. Performance testing is measured when the business gets peak by its hits. Another research was conducted on different tools to do performance testing [7]. Apache JMeter is a free java application performance testing tool. It has a lot of plugins to aid the testing tools. Performance testing using JMeter is a type of testing to determine the responsiveness, throughput, interoperability, reliability, and scalability of an

application under a given workload. In today's competitive world it has become critical to the organizations to test their web application [8]. Load testing is the process of subjecting a whole system to a work level approaching its limits. Load testing is done to determine the behaviour of the system under normal conditions and peak load conditions. The objective of load testing is to determine the maximum capacity of an application. Another research was done on web services used mostly in all aspects of social life [9]. For web applications, performance testing is gaining wide attention. This study highlights first we analyse and research the types, and methods of performance testing of the web and later we do some testing process methods.

A study was conducted that the performance tool was used to test web applications. This tool is used for performance, load and stress testing of web applications or websites [10].

Another research was conducted on web application performance testing which plays an important role in providing Quality of service [11]. This study highlights the performance testing of web applications using a reactive based framework which helps in reducing the cost and increases the efficiency of performance testing.

A performance testing framework for a rest-based web application was proposed [12]. This framework aims to provide software testers with an integrated process from test case design, test scripts, and test execution. Another research was conducted on ajax based web applications [13] which have gained more popularity since it brings the richness of desktop applications.

A research was conducted regarding challenges and experiences to identify a good solution for conducting performance testing on web applications [14]. There was another study conducted which described three open-source tools and compared their performance, usability and software requirements [12].

Testing web applications is nothing but finding errors [12]. This study highlights different performance testing tools and tried with JMeter to improve the performance of website or web application. There was another research conducted which provides the comparison of load testing tools (Sharma,2016). In this study the main load test tools available in the market and their advantages were discussed.

A study was conducted on performance testing concepts and comparative analysis of web applications [18]. The main objective of performance testing is not to identify bugs but to eliminate performance bottlenecks.

Performance testing and load testing are some of the means to evaluate web application performance [19]. This study highlighted the load test of web applications with JMeter using blaze meter in cloud-based load testing. Another study was conducted in comparison to tools [20]. In this research comparison of three different tools was compared with respect to response times.

All these studies highlight different aspects of performance testing. However, I believe that performance testing is very important for the success of the project. Hence in this project will perform performance testing on the torpedo7 website.

3. RESEARCH METHODOLOGY

Research methodology for performance testing of Torpedo website project has been discussed below.

3.1 Comparative Analysis of Performance Tools

Table 1 below shows the comparative analysis of performance tools. After doing comparative analysis selected JMeter as the best tool for my project. We have also selected JMeter since its open-source tool to use and have hands-on experience on JMeter.

CRITERIA	JMeter	LoadRunner	NeoLoad
Commercial License	Open Source Tool	Yes	Yes
Cost	Free	Expensive	Moderate
Launched By	Apache Foundations	HP	Neoload
Result Reporting	Minimum Support (Need additional plugins for graphs)	Maximum Support	Maximum Support
Browser Support	Support all browsers	Support all browsers	Support all browsers
Load Generation	Unlimited load GENERATION	Depending on the type of license	Depending on the type of license
Devise OS Support	Supports Android, Linux, Windows, Mac	Does not support Mac	Does not support Mac

Table 1:Comparitive analysis of performance tools

3.2.Selected Tools for Project

3.2.1 Apache JMeter

Apache JMeter is a most popular and open-source tool for testing the web applications which is best suited for the companies as they do not have to pay for it. It is 100% pure java application which is designed to load test functional behavior and measure performance of the application. It is specially designed for testing Web applications. Apache JMeter is used to test both on static and dynamic resources performance. It can be used to simulate a heavy load on the server to analyze overall performance under different load types. We can use to make a graphical analysis of performance test scripts under heavy load [3].

3.2.2 Blazemeter

Blazemeter is a Chrome extension that enables you to record, browse, upload and run. Using Blazemeter we can create proper test scripts and load scenarios. Blazemeter enables us to write test scripts using JMeter and user experience test scripts using Selenium. All we need is to write the test-scripts in Blazemeter, choose the number of load-engines and run the test. The system takes care of everything else. An unlimited number of load-engines are preconfigured and available at our disposal. Detailed graphical reports are generated during the load [2].

3.3.Selected Methodology for Project

A project run by Flipmind company works on the agile methodology for the development of web applications. In the company, the project is divided into sprints and each sprint runs for 1 week. Daily stand-up meetings and discussions will be held in the company to give a brief description of project progress and about the issues facing [5].

3.3.1 Workflow of JMeter

Figure 1 shows the workflow of JMeter. When we start to perform the test of an application, JMeter creates requests to target servers and simulates the number of users by sending requests to the target server. Once the server starts responding to the requests JMeter saves all the responses. By using the response JMeter collects data to calculate statistical information. Finally, based on this information JMeter prepares a report about the performance of the Application under test.

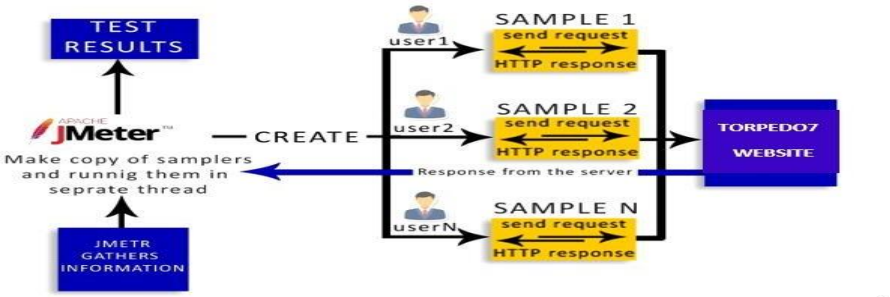


Figure 1: JMeter Workflow for Project

4 PROJECT EXECUTION

4.1 Test Plan

Table 2 below shows the test plan for my project. Table 2 is more focused on the test plan for the project of the Torpedo7 website and how I generated reports with different users.

Document	Performance Test Plan
Project	Torpedo7 website
Version	1.0.0
Objective The objective of this document is to outline the environment and Performance test Plan for the Torpedo7 website and to ensure if the current system architecture is optimal for handling increased traffic on the website and also, to figure out if there are any upgrades that need to be done.	
Scope • Load test the Homepage, Shop feature (Snow, Technology, Clothing, Water, Bike), Service feature (Bike workshop), Search functionality, adding the item to the cart and proceed to checkout functions of Torpedo7 website for 200 users and 500 users • Analyze test results. • Identify bottlenecks. • Propose suggestions.	
Out of Scope • Functional or accuracy testing of the Torpedo7 website. • Browser or software compatibility testing. • Any other testing types not included in the Scope section.	
Hardware Load Generator – Dell Intel Core i7 -3.40 GHz., 16GB RAM	

Software
Operating System – Windows 7 Professional

Performance Testing Tool
Apache JMeter 5.1.1

Environment

- Java Runtime Environment (JRE) is installed to run JMeter.
- JMeter is installed to create virtual users and do load testing.
- jpgc – standard set plugin is added to JMeter to generate more graphs.
- Chrome extension, Blaze Meter is added to chrome to record scripts.
- A good internet connection is Confirmed.

Risks
Torpedo7 is a live website. At the time of performance testing, it is not isolated. The load JMeter applying is not the only load for the web site. Also, at the same time, there will be a website getting requests from the outside world. So, the report analysis might be not very correct.

Approach and Execution Strategy Performance Test Script Steps
Performance Test scripts will be created using Blaze Meter. Those scripts will be developed to simulate the actions mentioned below.

- Torpedo7 website home page
- Shop feature
- Service feature
- Search
- Add to Cart
- Checkout

Performance Test Data Planning and Preparation

- Dynamic search data will be provided, and few searches will be done in one iteration.
- Parameters are added to search data.

Performance Test Scenario

- First, the Load Test will be performed with 200 threads with 3000 seconds ramp-up period for 3000 seconds of duration.
- Then the threads are increased to 500 with the same 3000 seconds ramp-up period for 3000 seconds of duration.

Performance Test Reports and Metrics • Built-in Listeners in JMeter are used to create Summary Reports, Graphs and Tables. • jp@gc – Standard Set plug-in will be added to the Test Plan to get more reports. • Metrics like Response Time, Deviation, Throughput, Min, Max, Error%, Latency will be tracked using these reports. • JMeter consumes less memory in Non-GUI mode which helps to generate CSV file and HTML Report in Windows **Execution and Analysis** • Load Test will be performed with 200,500 users. • Upgrades that need to be done will be identified.
Deliverables • Test Plan – This project report document • Test Results – Raw data captured from Performance Test execution • Final Test Report – Test Metrics with findings and suggestions.
Exit and Entry Criteria Entry Criteria: Functionally stable application. Data setup for the transactions**. Exit Criteria:** All the Performance objectives are met.

Table 2:Test plan for the Project

4.2 Performance Test Cases

The performance test cases for this project are shown in table 3. Table 3 test cases 2,13,14,19 and 20 are the main features of this torpedo7 website.

Test Case ID	Step Description	Expected Output	Transaction Name
TC_001	1. Open the Chrome browser2. Enter the URL 3. Enter the valid Email and Password4. Click the sign-in button	User should be landed on the Torpedo7 Login Page and should be logged in successfully	User login
TC_002	User click on Shop icon and navigate to snow page	Snow page is displayed	Snow page
TC_003	User click on Snowboarding and navigate to snowboarding page	Snowboarding page is displayed	Snow Boarding page
TC_004	User click on Shop icon and navigate to technology page	Technology page is displayed	Technology page
TC_005	User click on phone and navigate to the phone page	Phone page is displayed	Phone page
TC_006	User click on Shop and navigate to clothing and footwear	Clothing page is displayed	Clothing page
TC_007	User click on women's and navigate to women's page	Women page is displayed	Women page
TC_008	User click on Shop icon and navigate to the water	Water page is displayed	Water page

TC_009	User click on surfing in water page and navigate to surfing page	Surfing page is displayed	Surfing page
TC_010	User click on Shop icon and navigate to the Bike feature page	Bike page is displayed	Bike page
TC_011	User click on MTB-Hardtail and navigate to Hardtail mountain bike page	Hardtail mountain bike page is displayed	Hardtail page
TC_012	User click on Shop icon and navigate to Bike & Frames feature	Bikes & Frames features page is displayed	Bikes & Frames page
TC_013	User click on service icon and navigate to Bike workshop page	Workshop page is displayed	Workshop page
TC_014	User click on the Search button and search for shoes	Shoe page is displayed	Shoe page
TC_015	User click on the Search button and search for socks	Socks page is displayed	Socks page
TC_016	User click on the Search button and search for jackets	Jackets page is displayed	Jacket page
TC_017	User click on the Search button and search for boats	Boat page is displayed	Boat page
TC_018	User click on the Search button and search for bike	Bike page is displayed	Bike page
TC_019	Click on add to cart for the item user selected	Item has been added to cart	Add to cart
TC_020	Click on proceed to checkout	My details page should be displayed	Checkout
TC_021	The user enters the email id and clicks on the next page	Delivery and payment details page should be displayed	Checkout
TC_022	User fills the details first name, last name, phone number, and clicks on continue to delivery	Delivery options page should be displayed	Checkout
TC_023	User fills the address details and clicks on continue to payment	Payment details page should be displayed	Checkout
TC_024	User enter the credit card number, expiry date, name on card, CVC and click on complete purchase	Your credit card number is invalid	Checkout

Table 3:Performance Test Cases

4.3 Test Scripts

For accurate performance testing results, the test environment should be similar to the production environment. To achieve that, the whole application infrastructure should be thoroughly analyzed. Torpedo7 website which is subjected to the performance testing is already in the production environment since it is a live website. To avoid the external heavy load as much as possible, it is better to conduct the performance test after 10 p.m. as there will be fewer users accessing the Torpedo7 website at that time. Performance testing is conducted using JMeter to check the behavior of the Torpedo7 website on different load conditions. First, the performance test is conducted for 200 users with 3000 seconds ramp-up period for 3000 seconds.

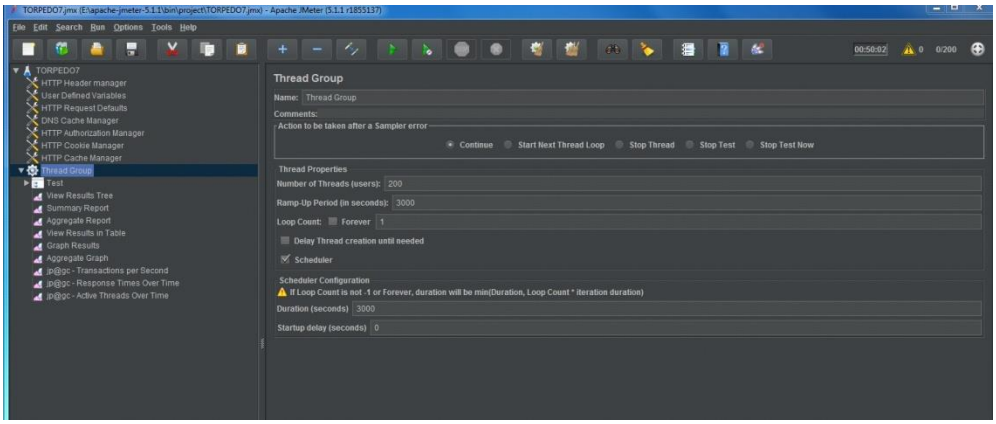


Figure 2: Test Scenario 1

Secondly, users were increased up to 500 and performance testing is conducted with 3000 seconds ramp-up period for 3000 seconds.

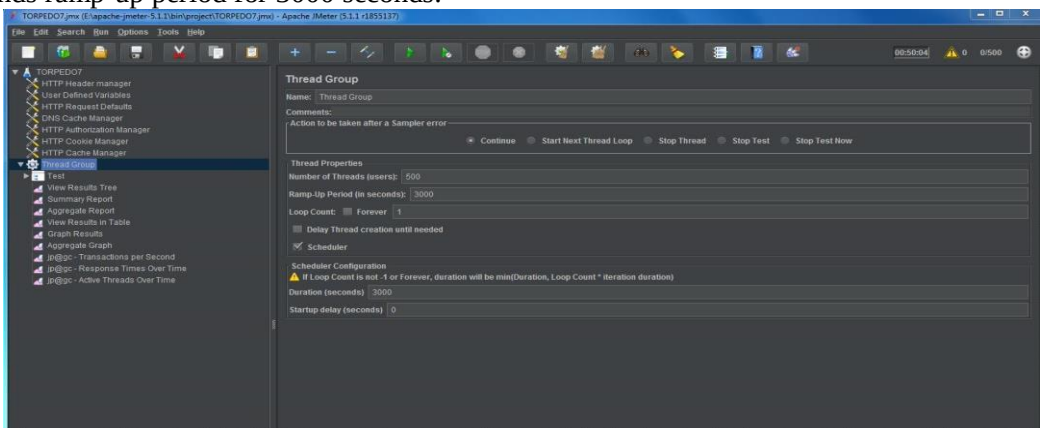


Figure 3: Test Scenario 2

Test Plan named TORPEDO7 was created to test the performance of the www.torpedo7.co.nz website using JMeter. Test scripts were recorded using Blaze Meter in the Chrome browser and exported to JMeter as JMX file.

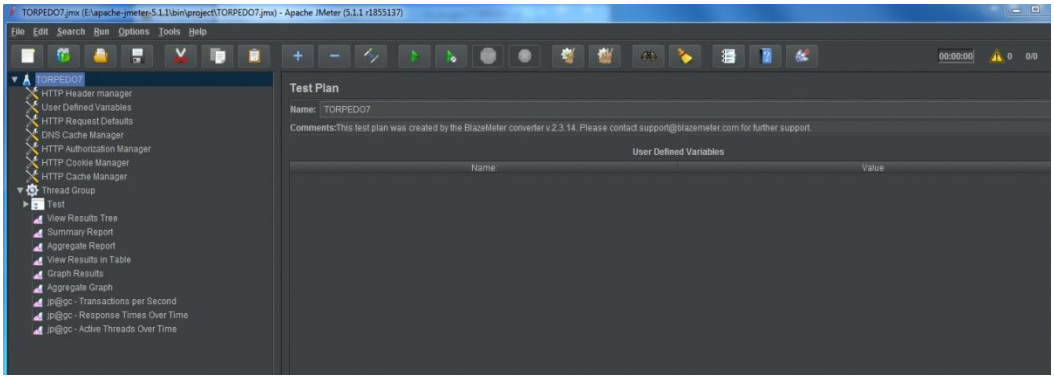


Figure 4: Test Plan

There are few Configuration Elements added to the Test Plan. Some of them are listed below:

1. HTTP Header Manager is added to the test plan at the Thread Group level which carries header information of requests.

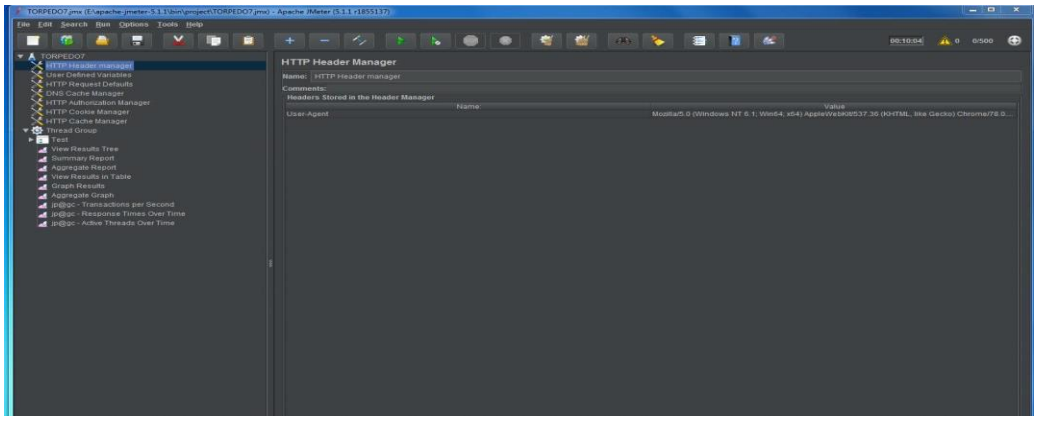


Figure 5: HTTP Header Manager

2. User-Defined Variables have been added to the Thread Group.

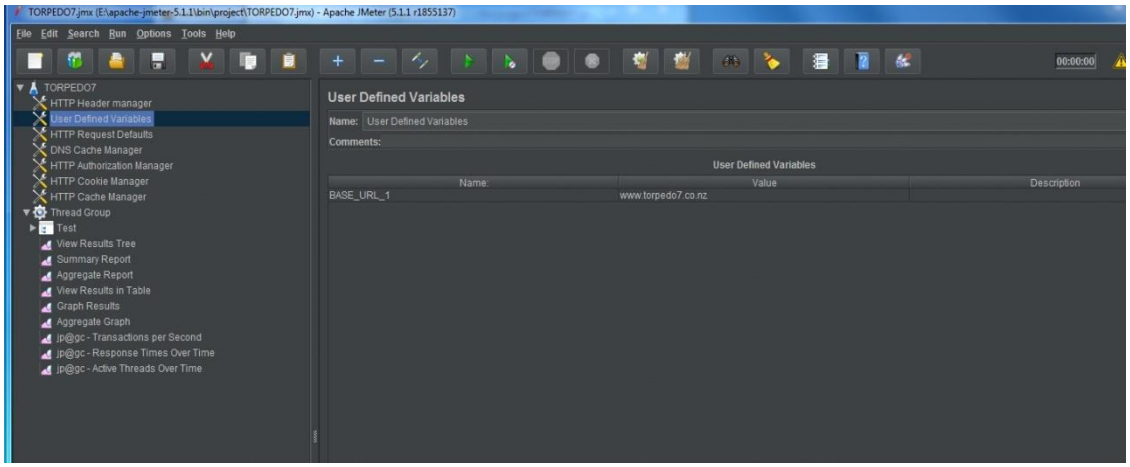


Figure 6: User-Defined Variables

3. HTTP Request Defaults have been added at the Thread Group level. Base URL is given there which is passed to every request.

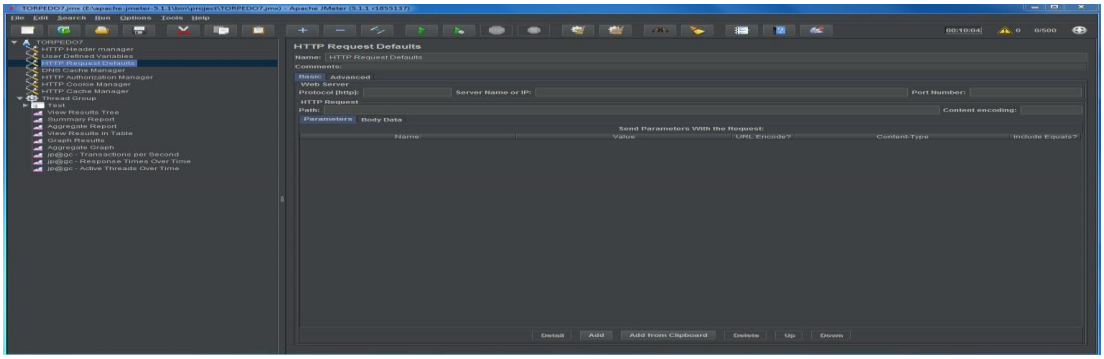


Figure 7: HTTP Request Defaults

4. DNS Cache Manager is added to the test plan at the Thread Group level. Clear cache each iteration option is selected to clear the DNS cache after each iteration.

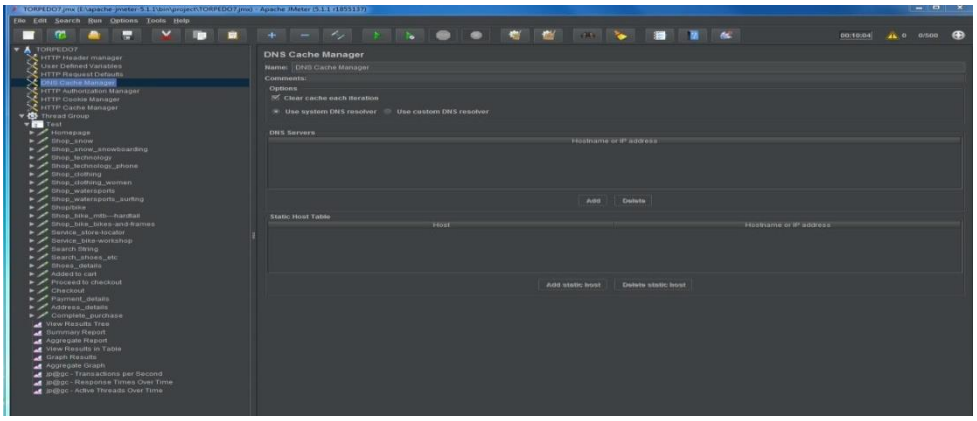


Figure 8: DNS Cache Manager

5. HTTP Cookie Manager is added at the Thread Group level. Clear cookies each iteration option is selected to clear HTTP Cookie data after each iteration.

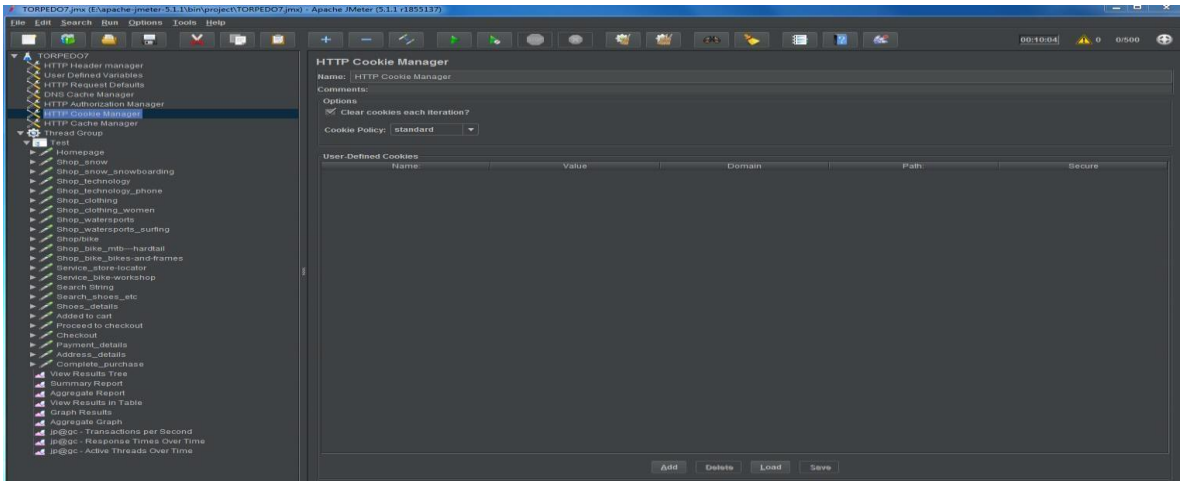


Figure 9: HTTP Cookie Manager

6. HTTP Cache Manager is added to the test plan at the Thread Group level. Clear cache each iteration option is selected to clear HTTP cache after each iteration and a maximum number of elements in the cache is set to 5000.

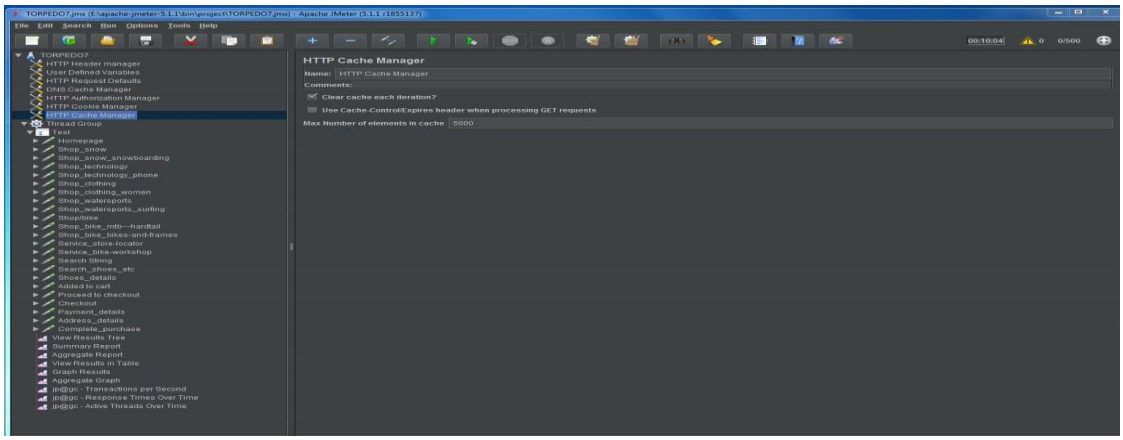


Figure 10: HTTP Cache Manager

Adding Parameters: Added parameters in Search String like shoes, socks, jackets, boats, bike

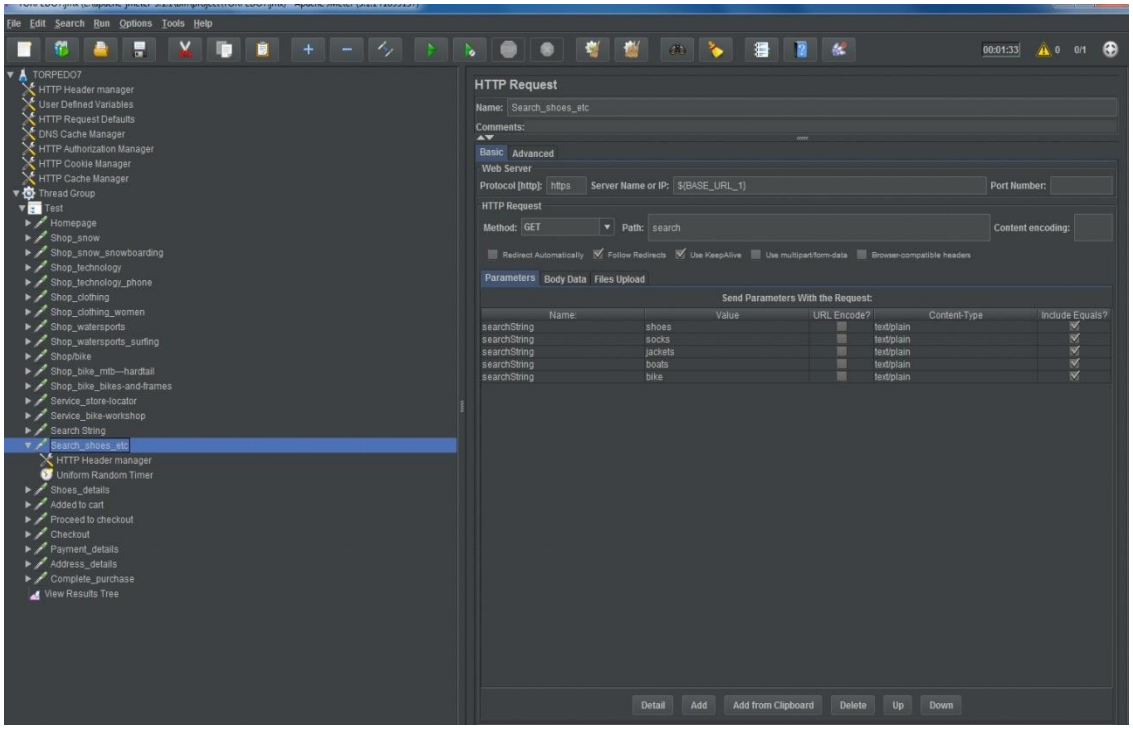


Figure 11: Adding Parameters

Generation of CSV file and HTML report: Using this command prompt generated CSV file for 200 users.

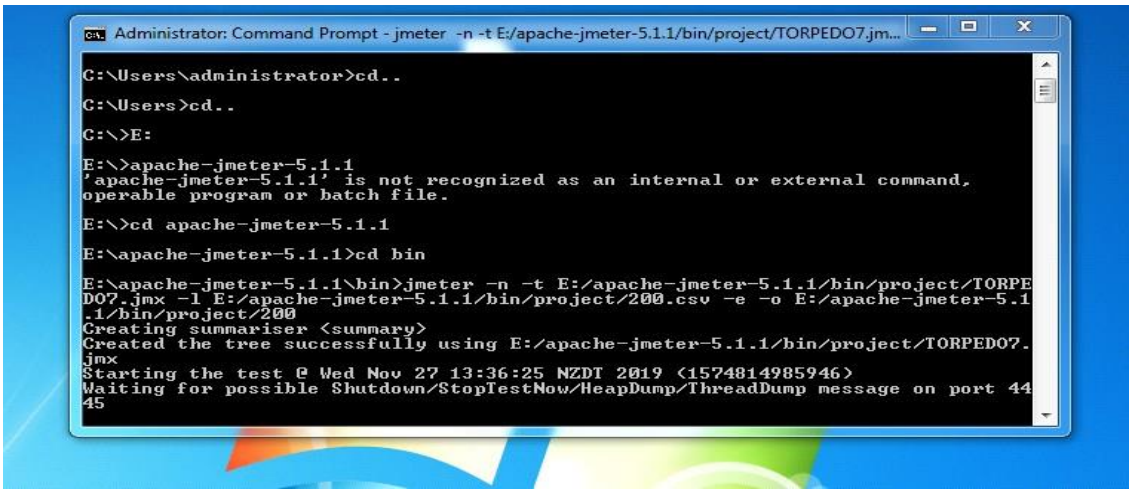


Figure 12: Command Prompt for 200 users

Generation of CSV file and HTML report: Using this command prompt generated CSV file for 500 users.

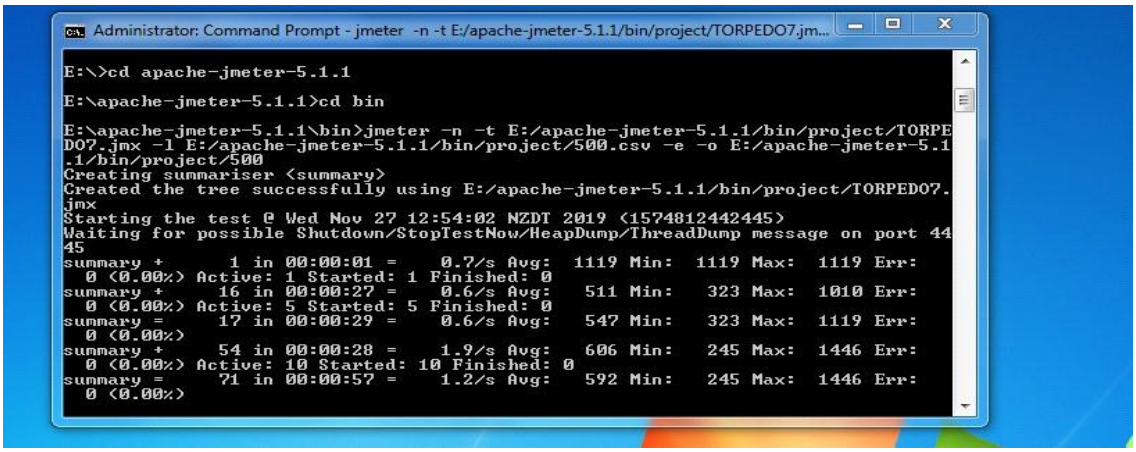


Figure 13: Command Prompt for 500 users

To create CSV file and HTML report in NON-GUI Mode we use the following command:

Jmeter -n -t E:/apache-jmeter-5.1.1/bin/documents/Homepage.jmx -l E:/apache-jmeter-5.1.1/bin/documents/test2.csv -e -o E:/apache-jmeter-5.1.1/bin/documents/htmlreport

Here

-n Non-GUI mode
-t location of Jmeter script
-l location of the result file
-e generate report dashboard after the load test
-o output folder for report dashboard

5 DISCUSSION

5.1 Analysis of Web Application Performance

5.1.1 Aggregate Report Comparison

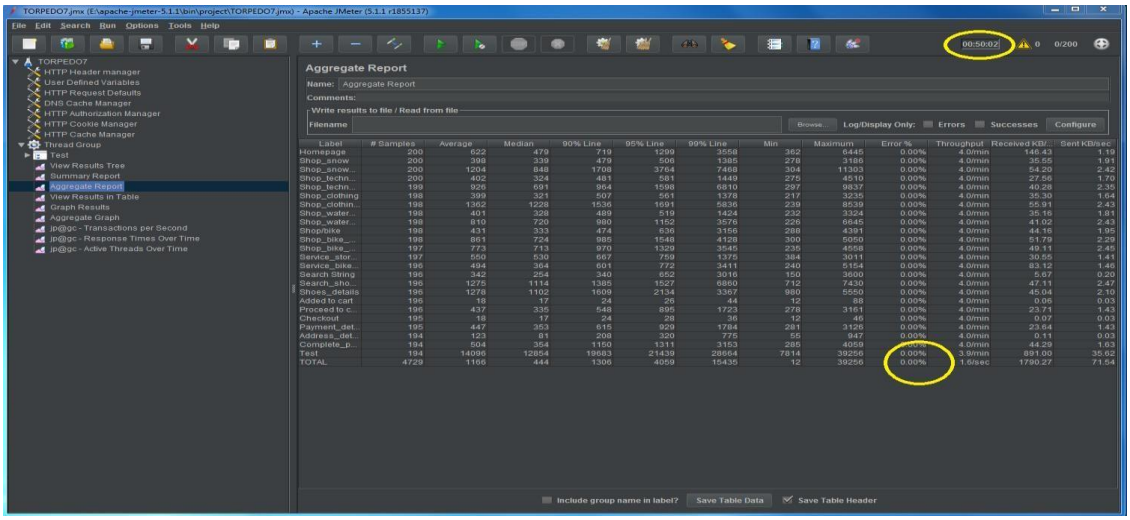


Figure 14: Aggregate Report (200 Threads)

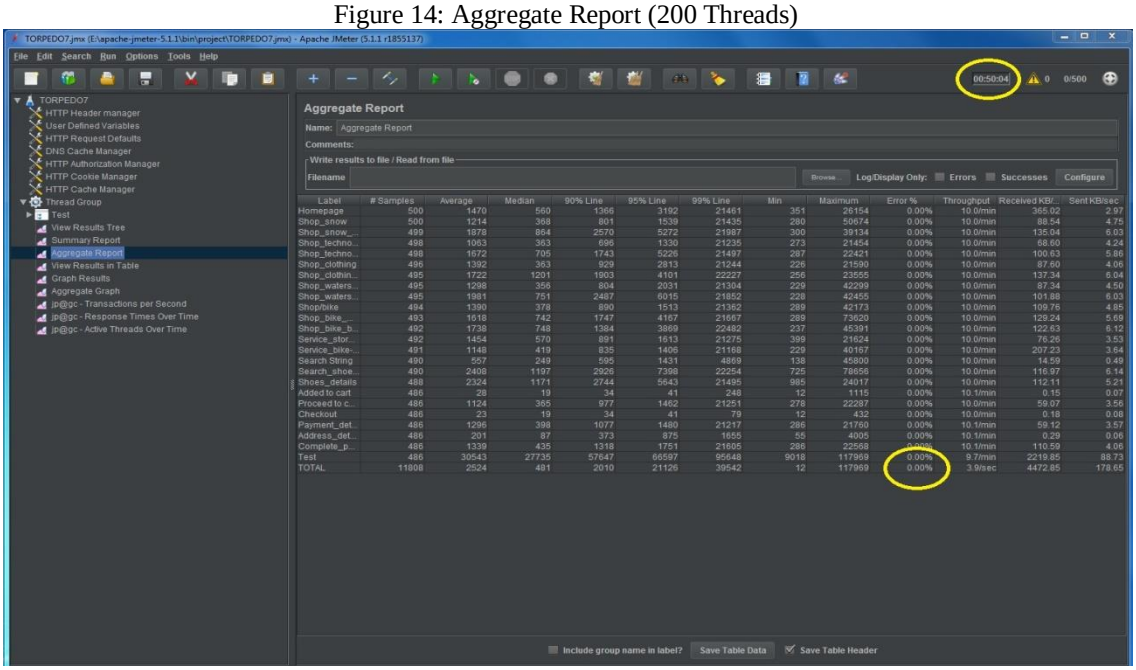


Figure 15: Aggregate Report (500 Threads)

- The aggregate report is the default report in JMeter.
- The aggregate report creates a table row for each differently named request in our test, which is similar to the summary report.
- We should mention the label names correctly to get the best results from the report.

- For both the scenarios, labels are the same as homepage, shop, service, add to cart and checkout.

- And when comparing samples, average, min, max, sent, received, std dev, throughput, avg bytes all values are different due to network issues.

- Errors for both the scenarios are showing 0% because all the tests are passed here.

- If anyone of the tests is failed, we get values in errors.

5.1.2 Aggregate Graph Comparison

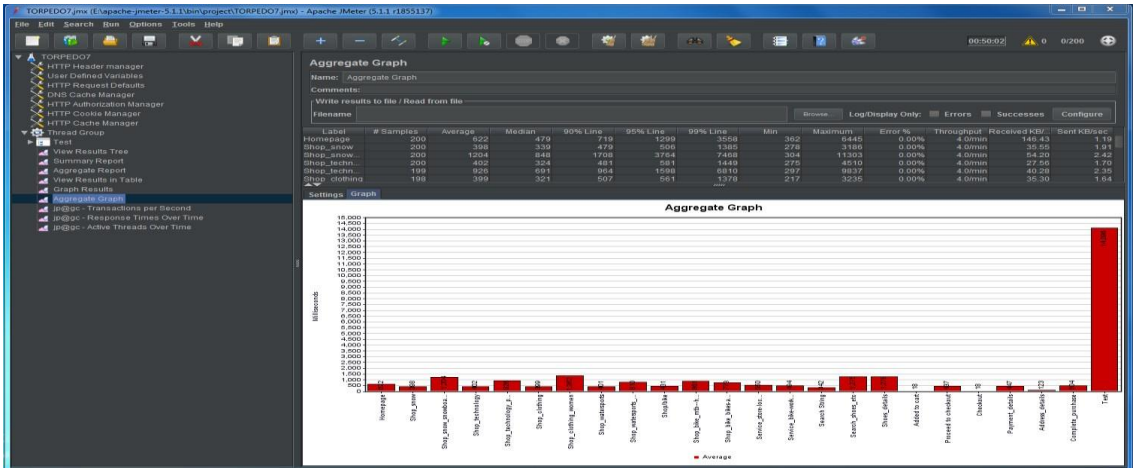


Figure 16: Aggregate Graph (200 Threads)

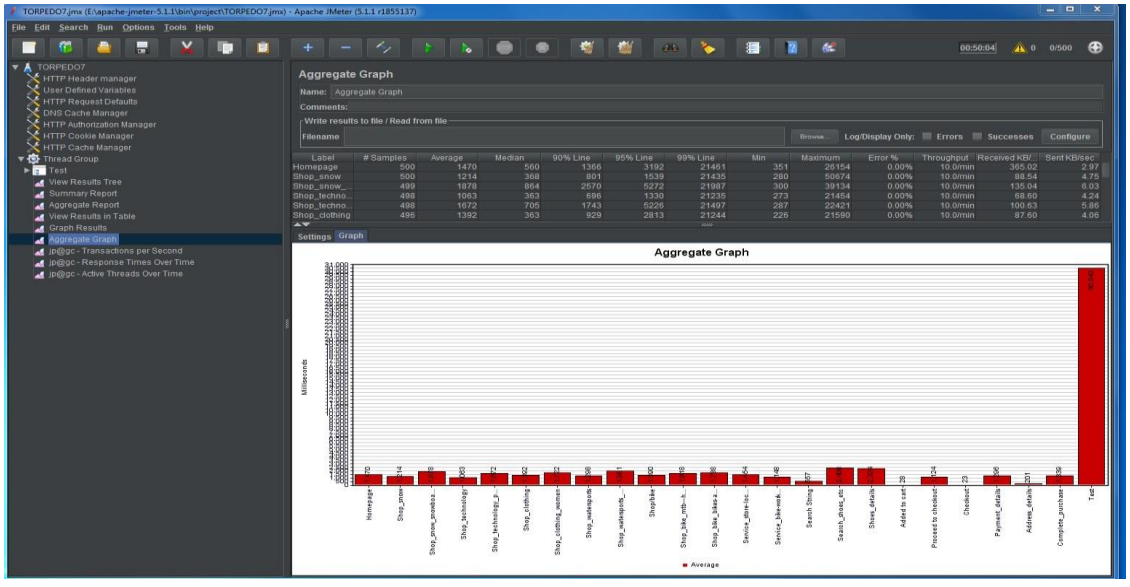


Figure 17: Aggregate Graph (500 Threads)

- The above figures compare the Average Response time and Median Response Time for every single request.

- According to the above two graphs, there is a huge difference between Average Response Time for 200 threads and for 500 threads for each request.

- There is also a huge difference between Median Response Time for 200 threads and for 500 threads for every single request.

- Finally, there is more deviation between Average Response Time and Median Response Time for each request.

5.1.3 Transaction per Second Graph comparison

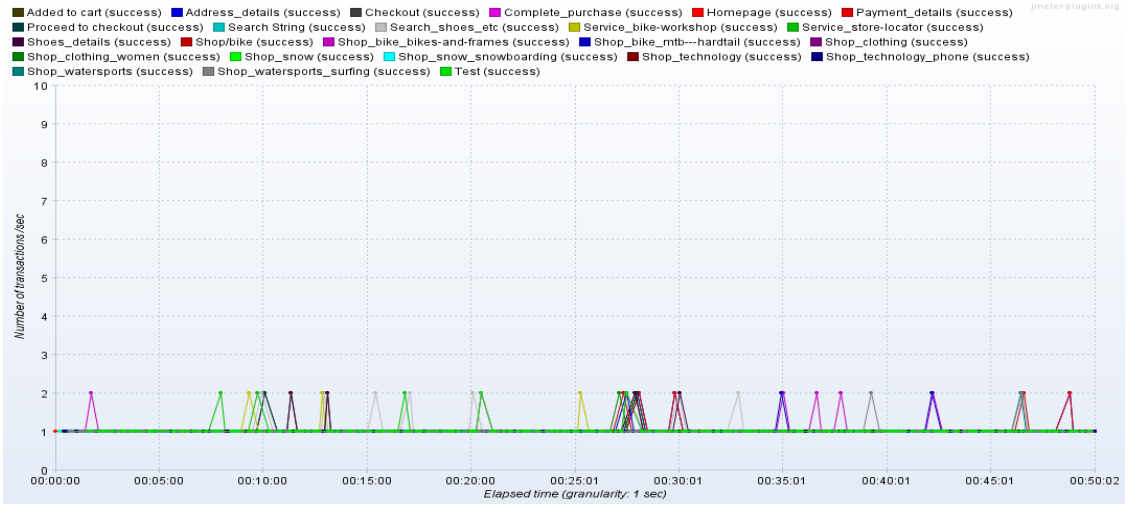


Figure 18: Transaction per Second Graph (200 Threads)

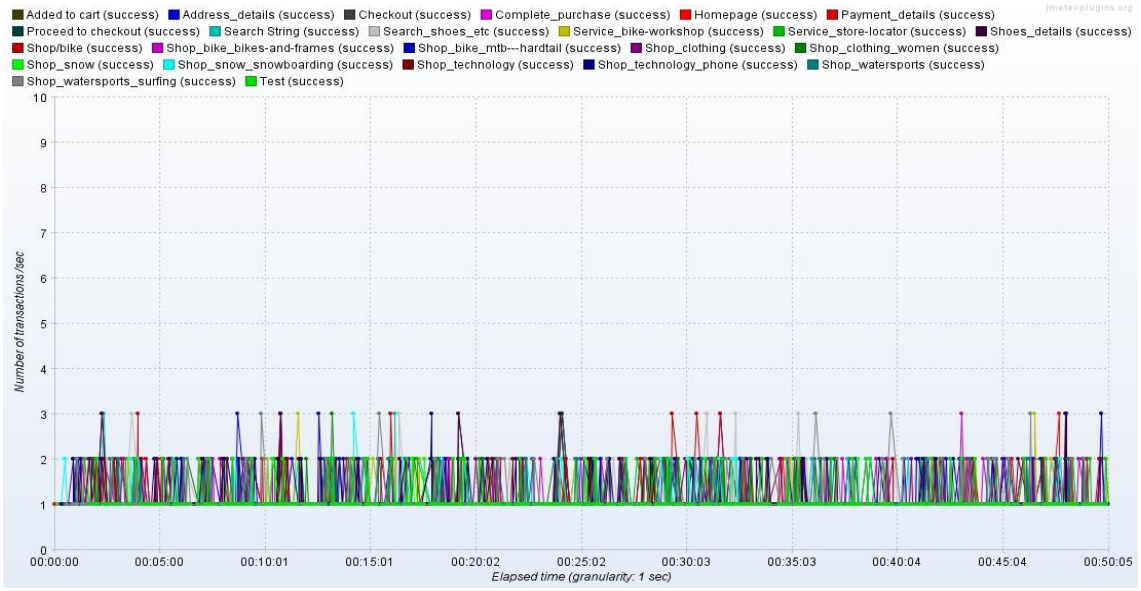


Figure 19: Transactions per Second Graph (500 Threads)

- The above figures 57,58 show the Transactions per Second for 200,500 Threads

.

- In figure 57 the transactions per second are 1 or 2 all the time.
- But when the threads increased from 200 to 500 transactions per second also increased mostly and reached to 3.

5.1.4 Graph Results comparison

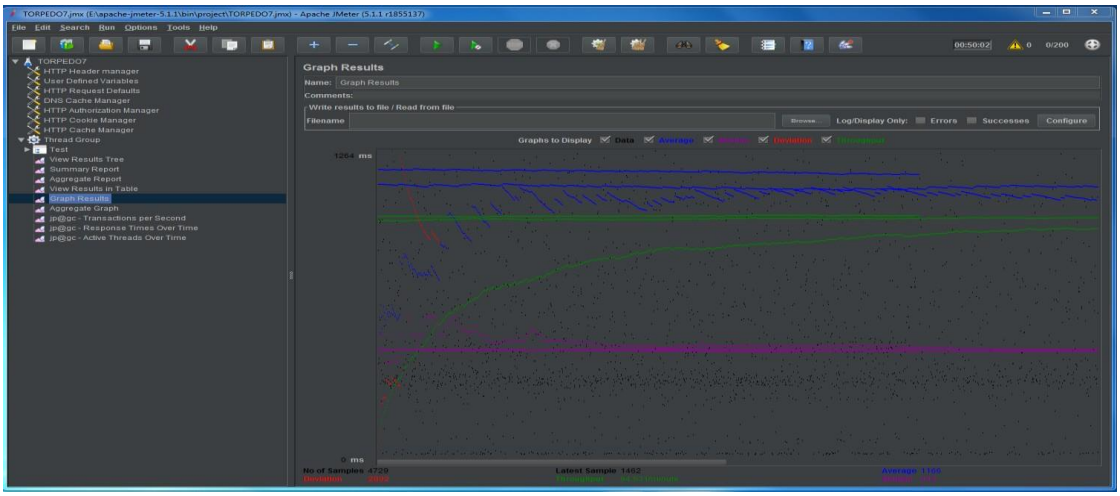


Figure 20: Graph Results (200 Threads)

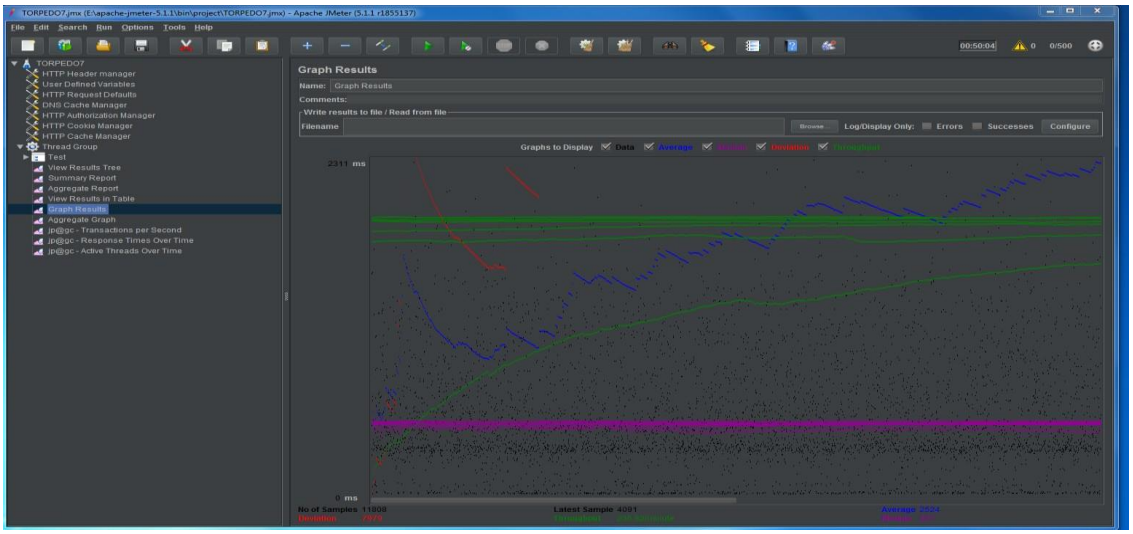


Figure 21: Graph Results Graph (500 Threads)

- This is the most important graph that shows the Median, Average response time, Deviation from the response time and Throughput.
- In figure 60 throughput gradually increased very high nearly two times compared to figure 59, because the first graph is for 200 threads and the second is for 500 threads.

	(200 Threads)	(500 Threads)
Throughput	**94.531/minute**	**235.82/minute**

- There is some difference between 200 threads and 500 threads in average response time, median and deviation.

	(200 Threads)	(500 Threads)
Average Response Time	**1166**	**2524**
Median	**444**	**481**
Deviation	**2892**	**7979**

In performance testing we start our script with recording, once we finish with recording, then we open our script and run. However, the first step we need to do in performance testing after recording is to identify the main major requests and delete all the redirected ones, otherwise we will end up with the issues frequently.

This research project report was generated with 200 users and 500 users, but when we started our test with 10 and 50 users we did not face any issues. But once we increased the test from 200 users to 500 users at the time of execution, we faced some issues then we performed brainstorming why are we getting issues for 200 users but not for 20,50 users?

In our research we came to know that in performance testing there are some issues, for example the server should be able to handle all token ids because each run creates freshly ids and server can note handle 500 requests. For this we slowly increased the number of threads and we observed firstly for 50 users then for 100 users so on and if errors were encountered after X numbers of users then it meant that the server was not ready to accept after that thread. We increased the number of users slowly like 20,30,40,50, 100 for 500 so on. Further it required more understanding. So, we increased the ramp-up time and duration and observed for 50 users at the beginning then increased the threads to 100 users then later to 200 and 500.

6. CONCLUSION

Torpedo7 website report covers a brief introduction about Flipmind Company and what they do in New Zealand and Torpedo7 website. Flipmind Company is using an agile methodology for its Torpedo7 website project. Torpedo7 website needs performance testing for Black Friday and Christmas deals. Using the JMeter tool created test scripts and analyzed the results. A load test is done for the Torpedo7 website using JMeter for the Homepage, Shop feature, Service feature, Search functionality, add to cart and checkout with 200, 500 users and hence analyzed the results using summary report, graphs, CSV file, and HTML report.

7. RECOMMENDATIONS

Torpedo7 website was tested with 200, 500 threads. With the analysis of test results, the current system is optimal for handling increased traffic on the website. Further, it does not need any upgrades now. The graphs will not remain the same every time because of the load stress or network issue. To get the proper and exact results it recommends executing the test night after 10 pm. Especially if working on the performance testing on live websites, it is always recommended to run the load test at night-time to get accurate results.

REFERENCES

[1] Patil, S. S., & Joshi, S. D. (2012). Identification of Performance Improving Factors for Web Application by Performance Testing. Int. J. Emerg. Technol. Adv. Eng., 2(8), 433-436.

[2] Arul, D. P., & Asokan, M. (2014). Load testing for query based e-commerce web applications with cloud performance Testing Tools. International Journal of Computer Engineering & Technology (IJCET), 5(10), 01-10.

[3] Chandel, V., Patial, S., & Guleria, S. (2013). Comparative Study of Testing Tools: Apache JMeter and Load Runner. International Journal of Computing and Corporate Research, 3(3).

[4] Khan, R., & Amjad, M. (2016). Performance testing (load) of web applications based on test case management. Perspectives in Science, 8, 355-357.

[5] Laanti, M., Salo, O., & Abrahamsson, P. (2011). Agile methods rapidly replacing traditional methods at Nokia: A survey of opinions on agile transformation. Information and Software Technology, 53(3), 276-290.

[6] Khan, R., & Amjad, M. (2016). Performance testing (load) of web applications based on test case management. Perspectives in Science, 8, 355-357.

[7] Erinle, B. (2013). Performance testing with JMeter 2.9. Packt Publishing Ltd.

[8] Arul, D. P., & Asokan, M. (2014). Load testing for query based e-commerce web applications with cloud performance Testing Tools. International Journal of Computer Engineering & Technology (IJCET), 5(10), 01-10.

[9] Zhu, K., Fu, J., & Li, Y. (2010). Research the performance testing and performance improvement strategy in web applications. In 2010 2nd international Conference on Education Technology and Computer (Vol. 2, pp. V2-328). IEEE.

[10] Van, P. H. O., Phipps, D. A., & Lin, S. (2016). U.S. Patent Application No. 14/730,692.

[11] Rasal, Y. M., & Nagpure, S. (2015). Web application: Performance testing using the reactive based framework. IJRCCT, 4(2), 114-118.

[12] Kao, C. H., Lin, C. C., & Chen, J. N. (2013). Performance testing framework for rest-based web applications. In 2013 13th International Conference on Quality Software (pp. 349-354). IEEE.

[13] Dhote, M. R., & Sarate, G. G. (2012). Performance testing complexity analysis on Ajax-based web applications. IEEE Software, 30(6), 70-74.

[14] Kiran, S., Mohapatra, A., & Swamy, R. (2015). Experiences in performance testing of web applications with Unified Authentication platform using Jmeter. In 2015 international symposium on technology management and emerging technologies (ISTMET) (pp. 74-78). IEEE

[15] Hussain, S., Wang, Z., Toure, I. K., & Diop, A. (2013). Web service testing tools: A comparative study. arXiv preprint arXiv:1306.4063.

[16] Patil, S. S., & Joshi, S. D. (2012). Identification of Performance Improving Factors for Web Application by Performance Testing. Int. J. Emerg. Technol. Adv. Eng., 2(8), 433-436.

[17] Sharma, M., Vaishnavi, S. I., Sugandhi, S., & Abhinandhan, S. (2016). A comparative study of load testing tools. International Journal of Innovative Research in Computer and Communication Engineering, 4(2), 1906-1912.

[18] Jha, N., & Popli, R. (2017). Comparative analysis of web applications using JMeter. International Journal of Advanced Research in Computer Science, 8(3).

[19] Arslan, M., Qamar, U., Hassan, S., & Ayub, S. (2015). Automatic performance analysis of cloud-based load testing of web-application & its comparison with traditional load testing. In 2015 6th

IEEE International Conference on Software Engineering and Service Science (ICSESS) (pp. 140-144). IEEE.

[20] Bhardwaj, S., & Sharma, A. K. (2015). Performance Testing Tools: A Comparative Analysis. International Journal of Engineering Technology, Management and Applied Sciences, 3(4).

AUTHORS

First Author

I am Sushma. I have completed my Bachelor's in Computers Science from JNTU University India. I have recently completed my graduate diploma in Software Testing from AGI Education Limited New Zealand. I am passionate about testing and love to deliver quality products to customers.

Second Author

Dr. Shahid Ali is a senior lecturer and IT program leader at AGI Education Limited, Auckland, New Zealand. He has published number of research papers on ensemble learning. His expertise and research interests include ensemble learning, machine learning, data mining and knowledge discovery.

A SELF-ATTENTIONAL AUTO ENCODER BASED INTRUSION DETECTION SYSTEM

Bingzhang Hu and Yu Guan

School of Computing, Newcastle University, Newcastle upon Tyne, UK

ABSTRACT

Intrusion detection systems (IDSs) have received increasing attention in recent years due to the rapid development of Internet applications and Internet of Things. Anomaly based IDSs are preferred in many situations due to their capabilities of detecting novel unseen attacks. However, existing works have neither considered the intrinsic relationships within the network traffic data nor the correlations shared among the sub features (i.e. content feature, host-based feature, etc.). In this paper, we propose a self-attentional auto-encoder based intrusion detection system, namely the STAR-IDS, to effectively explore the intrinsic structures of network traffic data and evaluated it on the NSL-KDD dataset. The experimental results show that the proposed STAR-IDS has achieved state-of-the-art performances.

KEYWORDS

Intrusion Detection System, Auto Encoder, Anomaly Detection, Self-attentional

1. INTRODUCTION

Intrusion detection has been a popular topic since the emerging of Internet techniques and has received increasing attention due to the monumental growth of the Internet applications and Internet of Things (IoTs) in the past few decades. Many intrusion detection systems (IDSs) have been proposed to assist the network administrators to detect those abnormal network traffic data which may threaten the computers or network security.

Among the existing IDSs, the very early of them are mostly based on expert systems, in which a list of expert-defined signatures and patterns are matched with the input network traffic data to detect attacks. However, it is never easy to keep the libraries of such signatures and patterns up to date, therefore the rule-based expert systems suffer from unknown attacks when the incoming network traffic data contain system-agnostic attack-related signatures and patterns. Taking advantage of machine learning techniques, anomaly detection based IDSs have been proposed recently to tackle the unknown attacks, where the intrusion detection is usually treated as a classification problem. In anomaly detection based IDSs, network traffics are identified as normal traffics or abnormal ones. For example, [1] proposed a recurrent neural networks based method to identify whether the network traffic is normal or anomalous and further classify abnormal traffics into four attack types: Denial of Service (DOS), User to Root (U2R), Probe (Probing) and Root to Local (R2L), in a supervised fashion. However, supervised methods rely on sufficient training data with annotated labels, which is labour and time consuming. Without requiring labelled data, [2] employed an auto encoder network to distinguish normal and anomalous. The auto encoder is trained only on normal network traffics to minimise the reconstruction error between input and output. Hence in the deployment, the reconstruction error

David C. Wyld et al. (Eds): ISPR, ARIN, MLCL, NBIoT - 2020
pp. 53-61, 2020. CS & IT - CSCP 2020 DOI: 10.5121/csit.2020.100704

can be compared with a threshold as the reconstruction errors of abnormal data are supposed to be higher than those of normal ones.

Although many efforts have been put on learning good models [3] [4] [1] and selecting more efficient features [5], there are very few works looking at the intrinsic relationships of network traffic data. Network traffic data contain prolific information covering the intrinsic, content, host-based and time-based features of network packages [6], which describe different characteristics of network traffic. The anomaly network traffic data can be different with normal ones either at a holistic level or at a sub level, for example, the attack `port scan' is mostly different with normal traffic on the number of TCP connection requests of different ports within a short time [7], or the combination of some sub levels. To effectively and sufficiently leverage such intrinsic relationships within the network traffic data to improve intrusion detection, in this paper we propose a **S**elf-a**T**tentional **A**uto encode**R** based anomaly detection framework, namely STAR-IDS. Different with existing auto encoder based methods that directly measure the reconstruction errors, in the proposed STAR-IDS firstly split the input network traffic data into four sub features, each of which will pass through an independent auto encoder to obtain their corresponding reconstructions. Simultaneously, the original network traffic data in their entireties are fed into a self-attentional module to compute a set of attention weights that are multiplied with the reconstructions to obtain the adjusted reconstructions. Finally, the mean square error (MSE) between adjusted reconstructions and original network traffic data is compared with a threshold, and the traffic data with reconstruction errors higher than the threshold are considered as anomalies.

The remainder of this paper is organized as follows. In section 2, we review the related works of IDSs, especially those based on the anomaly detection; In section 3, we introduce the proposed Self-aTtentional Auto encodeR Intrusion Detection System (STAR-IDS). In section 4, we evaluate the proposed STAR-IDS on a benchmark dataset, and we give a brief conclusion and outlook in section 5.

2. RELATED WORKS

Intrusion detection systems can be generally divided into three categories: Signature-based (knowledge-based) Detection (SD), Anomaly-based (behaviour-based) Detection (AD) andStateful Protocol Analysis (specification-based) (SPA) [8]. In signature-based detection systems, predefined unique patterns (*e.g.* a sequence of code, a pattern or string that corresponds to a known attack, the hash code of a known bad file, *etc.*) are compared with incoming network activities to detect the intrusions. These patterns are either defined heuristically or by domain experts. In anomaly-based detection methods, the anomalies, which are defined as those network activities which differ from others enough to raise suspicion, are detected as intrusions. For the stateful protocol analysis, the protocol states are known to the system thus the vendor-developed generic profiles to specific protocols can be utilised to distinguish intrusions. Despite the effectiveness and simpleness of SDs in detecting known attacks and the ability of identifying unexpected sequences of commands in SPAs, ADs are standing out because of their abilities in detecting unknown attacks. In this section, we mainly review the existing works that belong to anomaly-based intrusion detection.

Anomaly-based intrusion detection systems typically employ supervised machine learning techniques as their backends. For example, [3] proposed a model that combines the Random Tree and Naive-Bayes Tree to classify the incoming network traffic data into two classes: normal and abnormal traffic. To explore the effectivities of features, [9] performed a feature selection and then employed the support vector machine (SVM) to detect abnormal traffics. Inspired by natural language processing, network traffic data are treated as documents in [10] and classified K-

Nearest Neighbour (K-NN). Recently, deep neural networks, as an emerging powerful machine learning technique, have also attracted much attention from the intrusion detection community. [1] proposed a recurrent neural networks (RNN) based method for intrusion detection, where each network traffic data can be identified as normal or abnormal and then further classified into a specific category including DOS, U2R, Probing and R2L. Similarly, [4] employed long-short-term-memory (LSTM) for intrusion detection and discussed the impacts of a variety of neural network architectures.

However, the aforementioned supervised learning methods, especially the deep neural networks, rely on a large volume of annotated data, which is time and labour consuming to be obtained. To tackle this problem, unsupervised approaches have been utilised. For example, [11] proposed a framework combining the self-taught learning and MAPE-K framework to deliver a scalable, self-adaptive and autonomous intrusion detection system. [2] proposed an auto-encoder based method, in which the reconstruction error between decoded traffic data and original incoming traffic data is used to validate if incoming traffic is abnormal.

3. METHODS

3.1. Overview

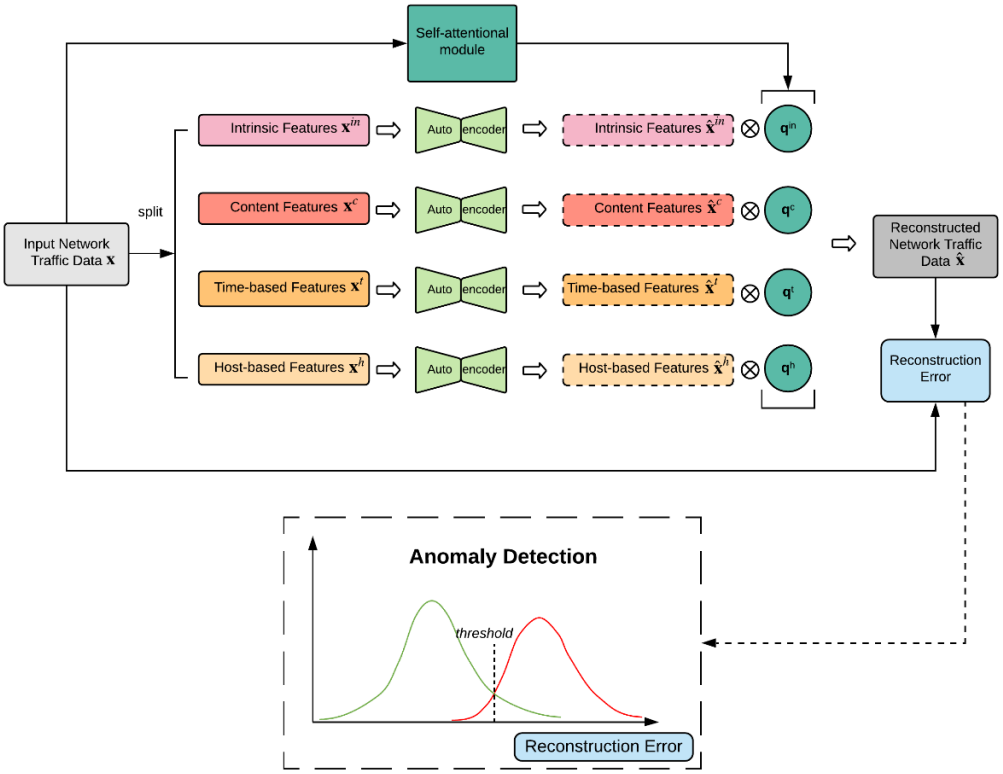


Figure 1. The framework of proposed STAR-IDS. The incoming network traffic data will be firstly split into four sub features, *i.e.* intrinsic features, content features, time-based features and host-based features. Each sub feature is encoded and decoded by an independent auto encoder.Simultaneously, the network traffic data is passed through a self-attentional module to obtain aset of attention weights, which is later multiplied with decoded sub features to get the adjusted reconstructed network traffic data. Finally, the reconstruction error is computed by measuring the l_2 loss between reconstructed and original traffic data, and the error is compared with a threshold to determine if the input traffic data is normal or abnormal.

The framework of proposed STAR-IDS is illustrated in Figure 1, from which we can see the input network traffic data, for example the i^{th} record $\mathbf{x}_i$, is first split into four subcategories: the intrinsic feature $\mathbf{x}_i^{in}$, which contains the basic information about the packet; the content feature $\mathbf{x}_i^c$, which covers the login trials, su attempts, number of roots, *etc.* information; the time-based feature $\mathbf{x}_i^t$, which records the traffic input over a two-second window; and the host-based feature $\mathbf{x}_i^h$, which shows the information over a series of connections. Then the sub features aresubsequently fed into four auto encoders to obtain their reconstructed vectors $\hat{\mathbf{x}}_i^{in}$, $\hat{\mathbf{x}}_i^c$, $\hat{\mathbf{x}}_i^t$and $\hat{\mathbf{x}}_i^h$. Simultaneously, the input network traffic data $\mathbf{x}_i$ is fed into a self-attentional module to compute four attentional weights $\mathbf{q}_i^{in}, \mathbf{q}_i^c, \mathbf{q}_i^t$ and $\mathbf{q}_i^h$ that are later on multiplied by the reconstructed feature vectors to get the adjusted reconstructed feature $\hat{\mathbf{x}}_i$. Finally, the reconstructed error is obtained between adjusted reconstructed feature and original input network traffic data and then compared with a threshold to determine if input data is anomaly. In the following subsections, we in turn introduce each component in the proposed STAR-IDS.

3.2. Auto Encoder

Auto encoder is a type of artificial neural network that is first proposed in [12] to learn internal representations by error-propagation. It is later employed in many machine learning frameworks to learn efficient representations of the input data in an unsupervised manner. The variations of auto encoder, including Sparse Auto Encoders [13], which impose the sparsity of the learnt embeddings; Denoising Auto Encoders [14], which aim to recover corrupted data by manually constructing corrupted-clean data pairs for training; Convolutional Auto Encoders [15], which introduce the convolution operations instead of linear mappings to learn the semantics; and *etc.*, have been widely used in anomaly based intrusion detection. However, none of the existing works have considered the intrinsic relations among the network traffic data and treat the sub features independently. Quoting the example in [7], if a large number of TCP connection requests to a very large number of different ports are observed within a short time, one could assume that someone is committing a `port scan'. Such kind of anomaly pattern can be found in host-based features. However, treating the network traffic data as an entirety may break such intrinsic structure. Therefore, to explicitly learn the semantic latent representations on each sub feature, we introduce four independent auto encoders to extract and learn the latent representations of each sub feature respectively. As shown in Figure 1, the sub features $\mathbf{x}^{in}$, $\mathbf{x}^c$, $\mathbf{x}^t$and $\mathbf{x}^h$ are fed into the corresponding auto encoders to obtain their reconstructed vectors as:

$$\hat{\mathbf{x}}^s = AE^s(\mathbf{x}^s), \tag{1}$$

where $s \in S = \{in, c, t, h\}$ denoting different sub features. The auto encoders in STAR-IDS

adopted the simplest fully connected architecture with a pre-defined number of hidden units, hence the mapping function AE can be written as:

$$AE^s(\mathbf{x}^s) = \sigma\left(W_{l^s}^s h_{l^s}^s + b_{l^s}^s\right),$$

$$h_l^s = \sigma(W_{l-1}^s h_{l-1}^s + b_{l-1}^s),$$

$$\cdots$$

$$h_1^s = \sigma(W_0^s \mathbf{x}^s + b_0^s), \tag{2}$$

Where σ denotes the activation function, $W\ and\ b$ are trainable weights and h are outputs of each layer. The details of the hyper-parameters and implementations of STAR-IDS can be found at https://github.com/u112358/STAR-IDS.

3.3. Self-attentional Adjusted Reconstruction

Comparing the summation of reconstruction errors between $\mathbf{x}^s$and$\hat{\mathbf{x}}^s$with a threshold, one can distinguish the anomaly from normal records. However, there exist some normal records with high reconstruction errors because they have never been seen by the auto encoder. We assume the intrinsic information in each sub features and the correlations between them in normal records are different from those of anomaly, and such kind of information can be utilised to adjust the reconstruction error. To this end, we proposed a self-attentional module, which is shown in Figure 1. Inspired by Spatial Transformer Networks (STN) [16], where a localisation net is employed to obtain a set of transformation coefficients that can recover the spatial manipulation of the input image, the self-attentional module adopts a regression network design whereby a set of attention weights is obtained from the input network traffic data and used to adjust the reconstructed sub features. Denoting the self-attentional module as f, the attentional weights can be written as $\mathbf{q} = f(\mathbf{x})$. Therefore, the adjusted reconstruction vectors can be obtained by:

$$\hat{\mathbf{x}} = [AE^s(\mathbf{x}^s) \otimes \mathbf{q}^s],$$

where $[\cdot]$ denotes the concatenation of each adjusted reconstructed sub features and $\otimes$ denotes the element-wise multiplication.

3.4. Anomaly Detection

After obtaining the adjusted reconstructed vector, the anomaly detection can be conducted by measuring the mean square error between adjusted reconstructed vector $\hat{\mathbf{x}}$ and the original incoming traffic data $\mathbf{x}$ and compare it with a pre-defined threshold δ. If the mean square error $e = \|\hat{\mathbf{x}} - \mathbf{x}\|_2^2$ is higher than the δ, then the incoming traffic data will be considered as an anomaly record and vice versa.

4. EXPERIMENTS

4.1. Dataset and Pre-processing

In this section, we evaluated the proposed STAR-IDS on the most popular NSL-KDD dataset, where the redundant records in KDD'99 [6] have been removed manually. Table 1 shows the statistic information of the NSL-KDD dataset, where we can see the proportion of normal and anomaly network traffic records in KDDTrain+ and KDDTest+ are very close. However, the distribution of each attack type (*i.e.* DoS, Probing, R2L and U2R) in KDDTrain+ and KDDTest+ are different. In KDDTrain+, the R2L and U2R attack records are minorities while in KDDTest+ the number of R2L records is close to that of Probing and occupies nearly a quarter of total records.

Table 1. Statistics of NSL-KDD Dataset.

	Abnormal				Normal	Total
	DoS	Probing	R2L	U2R		
KDDTrain+	45927	11656	995	52	67343	125973
KDDTest+	7458	2421	2754	200	9711	22544

The original network traffic records in NSL-KDD dataset are stored as 41-dimensional vectors, containing both numerical values and categorical values. To feed the data into proposed neural network, we converted the categorical values using one-hot encoding while the numerical values are normalised to range $[0,1]$. Hereby the final dimension of the data after pre-processing is 122.

4.2. Evaluations

4.2.1. Evaluation Metrics

Intrusion detection problem can also be regarded as binary classification (normal *vs.* anomaly) problem. In our experiments, four standard evaluation metrics, including accuracy, precision, recall and f-score are used to evaluate the performances of proposed method. The above-mentioned metrics are defined as:

$$Precision = \frac{\mathrm{TP}}{TP + FP},$$

$$Recall = \frac{\mathrm{TP}}{TP + FN},$$

$$F - score = \frac{2 \times Precision \times Recall}{Precision + Recall},$$

$$Accuracy = \frac{\mathrm{TP+TN}}{TP + FP + TN + FN}, \tag{3}$$

where TP (True Positive) denotes the correctly classified positive samples, FP (False Positive) denotes those negative samples classified as positive ones, TN (True Negative) denotes the correctly classified negative samples and FN (False Negative) denotes those positive samples classified as negative ones. It is important and necessary to consider various evaluation metrics at the same time as they can investigate the proposed method comprehensively hence illustrate us a complete picture of STAR-IDS.

4.2.2. Anomaly Detection on KDDTrain+

We first evaluated our proposed STAR-IDS on the KDDTrain+ Dataset. We followed the settings in [2] and conducted the training and testing process only on the KDDTrain+. The KDDTrain+ is randomly split into training, validation and test subsets, where the training subset contains 53,873 normal records only, while validation subset and test subset both contain 6,735 normal records and 6,735 anomaly records. Table 2 shows the detection performance between the proposed STAR-IDS and existing methods. We can find that among unsupervised methods, the proposed STAR-IDS has achieved better performances than [17] while yielded comparable

results with [2]. Regarding supervised methods [1] and [18], since the precision and recall rates are not given by the original papers, we leave them as absences. We can find although there is a gap between STAR-IDS and supervised methods, the results of STAR-IDS are also acceptable considering it using fewer data and no additional annotation information.

Table 2. Anomaly Detection Performances on NSL-KDD Dataset (KDDTrain+ only).

Methods	**Accuracy** (%)	**Precision** (%)	**Recall** (%)	**F-score** (%)
Auto Encoder [17]	93.62	91.39	96.33	93.80
De-noising AE [17]	94.35	94.26	94.43	94.35
AutoIDS [2]	96.45	**95.56**	**97.43**	96.49
STLNIDS [18]	98.30	n/a	n/a	**98.84**
RNN-IDS [1]	**98.81**	n/a	n/a	n/a
STAR-IDS (Ours)	95.59	95.26	95.77	95.51

4.2.3. Generalisation Abilities Analysis on KDDTest+

To further investigate the performance of proposed STAR-IDS, especially the generalisation abilities, we trained STAR-IDS on KDDTrain+ and evaluated it on KDDTest+ because of the various distribution in them. We compared the proposed STAR-IDS with several state-of-the-art methods and the results are shown in Table 3. We can find the proposed STAR-IDS has yielded the best performance among all the methods, even the supervised methods. Considering the results shown in Table 2 and Table 3, we can find the proposed STAR-IDS is stronger in detecting unseen abnormal traffic data, given that some records in KDDTest+ are novel attacks that never appears in KDDTrain+. Also, combining the results in Table 2 and Table 3, we can conclude that the STAR-IDS has better generalisation abilities and supervised methods, as well as AutoIDS, may overfit on the training set.

Table 3. Anomaly Detection Performance on NSL-KDD Dataset (KDDTrain+ & KDDTest+).

Methods	Accuracy (%)
Random Tree [3]	88.46
Random Tree and NBTree [3]	89.24
RNN-IDS [1]	83.28
DCNN [4]	85.00
STLNIDS [18]	88.39
LSTM [4]	89.00
Auto Encoder [17]	88.28
AutoIDS [2]	90.17
STAR-IDS (Ours)	**91.31**

5. CONCLUSIONS

In this paper, we proposed a self-attentional auto encoder based intrusion detection system, which takes the intrinsic relationships within the network traffic data as well as the correlations

between sub features of traffic data into account. Systematic experiments have been conducted and shown the proposed STAR-IDS is efficient and robust. However, there are still spaces to be improved in our work in the future. Firstly, the performances on the KDDTrain+ dataset show that the STAR-IDS is moderate, although the reason could be those methods are overfitted. Secondly, since the anomaly detection in STAR-IDS is based on the threshold, it will be interesting to discuss the impacts of the threshold on the performance. Finally, in the experiments, we have observed that some normal network traffic data get high reconstruction errors. It is worth to further explore that if the threshold-based STAR-IDS can be utilised for outlier detection.

ACKNOWLEDGEMENTS

This work is supported by the Engineering and Physical Sciences Research Council (EPSRC) Project CRITiCaL: Combatting cRiminals In the CLoud (EP/M020576/1).

REFERENCES

[1] C. Yin, Y. Zhu, J. Fei and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," Ieee Access, vol. 5, p. 21954–21961, 2017.

[2] M. Gharib, B. Mohammadi, S. H. Dastgerdi and M. Sabokrou, "AutoIDS: Auto-encoder Based Method for Intrusion Detection System," arXiv preprint arXiv:1911.03306, 2019.

[3] J. Kevric, S. Jukic and A. Subasi, "An effective combining classifier approach using tree algorithms for network intrusion detection," Neural Computing and Applications, vol. 28, p. 1051–1058, 2017.

[4] S. Naseer, Y. Saleem, S. Khalid, M. K. Bashir, J. Han, M. M. Iqbal and K. Han, "Enhanced network anomaly detection based on deep neural networks," IEEE Access, vol. 6, p. 48231–48246, 2018.

[5] S. Aljawarneh, M. Aldwairi and M. B. Yassein, "Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model," Journal of Computational Science, vol. 25, p. 152–160, 2018.

[6] M. Tavallaee, E. Bagheri, W. Lu and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in 2009 IEEE symposium on computational intelligence for security and defense applications, 2009.

[7] M. H. Bhuyan, D. K. Bhattacharyya and J. K. Kalita, "Network anomaly detection: methods, systems and tools," Ieee communications surveys & tutorials, vol. 16, p. 303–336, 2013.

[8] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin and K.-Y. Tung, "Intrusion detection system: A comprehensive review," Journal of Network and Computer Applications, vol. 36, p. 16–24, 2013.

[9] P. Kushwaha, H. Buckchash and B. Raman, "Anomaly based intrusion detection using filter based feature selection on KDD-CUP 99," in TENCON 2017 - 2017 IEEE Region 10 Conference, 2017.

[10] Y. Liao and V. R. Vemuri, "Use of k-nearest neighbor classifier for intrusion detection," Computers & security, vol. 21, p. 439–448, 2002.

[11] D. Papamartzivanos, F. G. Mármol and G. Kambourakis, "Introducing deep learning self-adaptive misuse network intrusion detection systems," IEEE Access, vol. 7, p. 13546–13560, 2019.

[12] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning internal representations by error-propagation," in Parallel Distributed Processing: Explorations in the Microstructure of Cognition, MIT Press, Cambridge, MA, 1986, pp. 318-362.

[13] A. Ng and others, "Sparse autoencoder".

[14] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," Journal of machine learning research, vol. 11, p. 3371–3408, 2010.

[15] J. Masci, U. Meier, D. Cireşan and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in International conference on artificial neural networks, 2011.

[16] M. Jaderberg, K. Simonyan, A. Zisserman and others, "Spatial transformer networks," in Advances in neural information processing systems, 2015.

[17] Aygun, R. Can and Y. A. Gokhan, "Network anomaly detection with stochastically improved autoencoder based models," in IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud, 2017.

[18] A. a. N. Q. a. S. W. a. A. M. Javaid, "A deep learning approach for network intrusion detection system," in Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies, 2016.

[19] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, S. Y. Philip and others, "Top 10 algorithms in data mining," Knowledge and information systems, vol. 14, p. 1–37, 2008.

[20] N. Sultana, N. Chilamkurti, W. Peng and R. Alhadad, "Survey on SDN based network intrusion detection system using machine learning approaches," Peer-to-Peer Networking and Applications, vol. 12, p. 493–501, 2019.

[21] S. Mukkamala, G. Janoski and A. Sung, "Intrusion detection using neural networks and support vector machines," in Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290), 2002.

[22] S. Mukkamala and A. H. Sung, "Detecting denial of service attacks using support vector machines," in The 12th IEEE International Conference on Fuzzy Systems, 2003. FUZZ'03., 2003.

[23] P. Mishra, V. Varadharajan, U. Tupakula and E. S. Pilli, "A detailed investigation and analysis of using machine learning techniques for intrusion detection," IEEE Communications Surveys & Tutorials, vol. 21, p. 686–728, 2018.

[24] E. Hodo, X. Bellekens, A. Hamilton, C. Tachtatzis and R. Atkinson, "Shallow and deep networks intrusion detection system: A taxonomy and survey," arXiv preprint arXiv:1701.02145, 2017.

[25] D. Heckerman, "Bayesian networks for data mining," Data mining and knowledge discovery, vol. 1, p. 79–119, 1997.

[26] N. Chand, P. Mishra, C. R. Krishna, E. S. Pilli and M. C. Govil, "A comparative analysis of SVM and its stacking with other classification algorithm for intrusion detection," in 2016 International Conference on Advances in Computing, Communication, & Automation (ICACCA)(Spring), 2016.

[27] D. H. Ballard, "Modular Learning in Neural Networks.".

[28] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning internal representations by error-propagation," in Parallel Distributed Processing: Explorations in the Microstructure of Cognition., vol. 1, MIT Press, Cambridge, MA, 1986, pp. 318-362.

AUTHOR INDEX